Strategy Instruction in Early Childhood Math Software: Detecting and Teaching Single-digit Addition Strategies

Kara Kilmartin Carpenter

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2013

ABSTRACT

Strategy Instruction in Early Childhood Math Software:

Detecting and Teaching Single-digit Addition Strategies

Kara K. Carpenter

In early childhood mathematics, strategy-use is an important indicator of children's conceptual understanding and is a strong predictor of later math performance. Strategy instruction is common in many national curricula, yet is virtually absent from most math software. The current study describes the design of one software activity teaching single-digit addition strategies. The study explores the effectiveness of the software in detecting the strategies first-graders use and teaching them to use more efficient strategies. Instead of a business-as-usual control group, the study explores the effects of one aspect of the software: the pedagogical agent, investigating whether multiple agents are more effective than a single agent when teaching about multiple strategies. The study finds that while children do not accurately report their own strategies, the software log is able to detect the strategies that children use and is particularly adept at detecting the effective use of an advanced strategy with a model that performs 67% better than chance. Overall, children improve in their accuracy, speed, and use of advanced strategies. Of the three teaching tools available to the children, the count on tool was most effective in encouraging use of an advanced strategy, highlighting a need to revise the other tools. Low-performers correctly used advanced strategies more frequently across the six sessions, while mid-performers improved after just one session and high-performers' correct use of an advanced strategy was consistent across the sessions. Whether a student saw lessons featuring a single agent or multiple agents did not have strong effects on performance. More

research is needed to improve the strategy detection models, refine the tools and lessons, and explore other features of the software.

Table of Contents

List of Tables

List of Figures

Acknowledgements

This dissertation is the result of countless great conversations with many inspiring individuals. First, I must thank my advisor, Herb Ginsburg, who has been a great source of questions, ideas, and support throughout my time at Teachers College. Additionally, I am grateful to Ryan Baker for introducing me to the world of big data in education. I also thank the other members of my committee for their guidance: Charles Kinzer, Janet Metcalfe, and Elizabeth Tipton. I thank the MathemAntics team: Ama Awotwi, Azedah Jamalian, Barron Ng, Ben Friedman, Dana Pagar, Esther Yoon, Rachael Labrecque, Rachel Kenny and Sam Creighan for their help in designing and implementing this study. I also thank the researchers who helped me collect data including Carissa Jung, Julia Xing, and Kristen McGregor. Finally, I thank the people who made this endeavor achievable: my parents, who encouraged me to continue my education, Kate, who listens patiently to often long-winded explanations of my research, and to Chad, who has supported me wholeheartedly. Thank you, all.

For 'lil nut

In early childhood mathematics, strategy-use is an important indicator of children's conceptual understanding and is a strong predictor of later math performance. Several children may each answer 5 + 7 correctly, but the way they got to that answer may differ. One boy may count five on one hand, then count up to five on the other hand switching to touching his fingers to his mouth for the final two. Another may start counting at 7 and continue counting 5 more. One girl may think to herself that 7 + 3 = 10 and then add two more. These differences reveal much more about a child's understanding and ability than simply knowing that all three answered correctly. Differences in strategy use may signal a math learning disability (Geary, Hamson, & Hoard, 2000) because children with such disabilities continue to use inefficient strategies much longer than their peers. Strategy use also predicts later math performance (Jordan, Hanich, & Kaplan, 2003; Siegler, 1998). Instruction about strategy is common in many national curricula, including *Everyday Mathematics* (2007) and TERC *Investigations* (2008), yet it is virtually absent from children's math software.

This paper describes the design of one software activity teaching single-digit addition strategies and a study conducted to explore the effectiveness of the software in detecting the strategies first-graders use and teaching them to use more efficient strategies. Specifically, the study addresses the following research questions: 1) What are the effects of using the math software on student's accuracy and strategy development? and 2) When teaching about four different strategies, is it more effective to have four separate pedagogical agents or a single agent explain the content? 3) How well do students assess their own strategy usage? 4) How well is the computer log able to detect what strategies students are using? and 5) What features in the computer log are particularly important in the detection of students' observed strategy?

In developing this study, I draw on disparate areas of research: educational software research, strategy variation literature, research on the development of mathematical thinking, and the emerging field of behavior detection using computer logging systems. Each of these areas of research influenced the design of both the software itself as well as the current study, from thinking about *microworlds* as a model for developing math software to examining how *learner-centered design* influenced the development of specific tools. Looking at software design from a gaming perspective also helped inform decisions about leveling and narrative structure, while the specific research on pedagogical agents led to questions about how multiple agents might increase the narrative elements and potentially improve learning outcomes. In designing the specific visual models used in the software, I drew heavily on the extensive research on addition strategy development, building conceptual understanding in mathematics, and *metastrategic* thinking. Finally, research on behavior detection gave insight into how to incorporate analysis of the software log into microgenetic research.

Theoretical Introduction

Educational Software Research

Most early childhood educational math software falls into one of two categories: 1)"edutainment," wherein digital bells and whistles predominate over minimal content or 2) discovery-based environments, which appear to offer deep content but do not present specific goals and objectives, leading to children using only superficial features (Sarama & Clements, 2002). A classic example of edutainment is Math Blaster, a game in which children need to shoot the correct answer of an operation using a space ray gun. Games like these offer little more than math fact practice and drill, a worksheet in game format that supports recall and procedural fluency, yet not building conceptual understanding. Much of this software claims to be

"research-based" on the basis of extremely limited summative research, rather than reflecting current cognitive theories (Clements & Battista, 2000).

Microworlds. On the other hand, software that is designed with a deep understanding of cognitive research and strong content knowledge can be very effective (Clements & Battista, 2000). Papert (1993) envisions creating "microworlds" with carefully designed logic, constraints, and interactions that reveal the mathematical content as children explore and experience the underlying structure. Microworlds based in Logo help elucidate the underlying structure of geometry and logic through the programming language. Children type instructions for the Logo turtle to follow, for example, telling it to turn right x degrees and walk forward y paces in an infinite loop. On a superficial level, this creates a fascinating drawing, whereas on another level children discover how manipulating the instructions creates different polygons, throwing light on deeper geometrical concepts. Other microworlds such as Cabri Geometry™ use drop-down menus and buttons rather than written commands to reveal the underlying structure of geometry, yet both share important structural and functional characteristics (Edwards, 1998). Structural elements of microworlds include computational objects that reflect content structure, linked representations, and goal-driven activities, while functional elements include manipulating objects, interpreting feedback, and creating solutions. Edwards argues that these elements encompass a wide range of both computer-based and physical materials, although the computer-based microworlds offer more direct feedback. Such feedback supports the process of debugging, in which by attempting to fix their mistakes, children discover the underlying structure of the microworld, although learning to debug may require explicit instruction in addition to exposure to the microworld (Carver & Klahr, 1986)

Technological affordances. Designing a microworld requires a deep understanding of the mathematical content the new world will model or teach, but also necessitates a deep understanding of the possibilities technology affords. Software should allow children to engage in significant learning through special activities not possible with traditional methods. Hoyles and Noss (2009) identify four categories of software tools that have the power to shape mathematics learning: 1) interactive graphical tools; 2) powerful information processing and data manipulation; 3) useful representational models and 4) digital communication. In illustrating their point, Hoyles and Noss draw on examples from a wide age range, yet each of these four categories is relevant to designing technology for the young math learner as well.

First, technology affords the ability to create interactive graphical tools. Sarama and Clements (2002) designed a series of interactive geometry activities for young children in their Building Blocks curriculum. The software lets children rotate, flip, slide, duplicate, combine, and deconstruct shapes to solve increasingly difficult challenges. Students complete the same tasks both with physical manipulatives and on the computer. Sarama and Clements find that when using the computer, students develop "compositional imagery" to figure out how many more shapes they will need. By eliminating the need for physical manipulation and increasing the accuracy of shape placement, the software allows the child to focus on the mathematics of shape building rather than the physical constraints of adjustment and fit.

The recent trend towards stealth assessment in gaming offers the potential to capture information about what children understand while they are playing games and report that information back to teachers and parents (Shute, 2011). Researchers are discovering new ways to use computer data logs to make complex inferences that offer more than simply accuracy and response time reports (Baker & Yacef, 2009). Children can also indirectly benefit from the

powerful information processing that computers provide their teachers. Using the mCLASS: Math™ system teachers conduct clinical interviews and record children's solutions, strategies, and mathematical explanations on hand held devices (Lee, Pappas, Chiong, & Ginsburg, 2010). Teachers use these formative assessments to guide their instruction. The information gleaned from these digital clinical interviews provides a more reliable assessment of students' abilities and better informs instruction when compared to popular curriculum-based measure assessments (Ginsburg, Chiong, Lee, & Pappas, 2010).

Technology also supports useful representational models not possible in traditional print media. One commercial example, MotionMath,™ provides children with a number line that zooms in to reveal more and more precise decimal notation and zooms back out to whole number notation. In order to place the number 5, the child needs to be at the whole number level, but in order to place the number .5, the child needs to zoom in to the tenths level between 0 and 1, and in order to place .05, the child zooms in further to the hundredths level between 0 and .10. Children need to zoom in and out of the number line in order to find the correct placement of numbers that fall from bubbles. They are able to zoom in and out quickly and accurately, playing with a visual model of a concept that used to be exclusively in the domain of mental model.

Digital connectivity offers opportunities for sharing and communication. Bottino & Chiappini (2002) describe how elementary students communicate mathematics solutions using their ARI-LAB system. Elementary school students use digital tools to solve math problems and then create solution sheets that are shared with classmates and teachers. The solution sheets support meta-level thinking by requiring students to select pertinent information from the workspace to copy into the solution space and writing notes to explain their work. Students can look at other students' solutions as an example, but may not copy and paste the others' work, and

must instead recreate the solution themselves. Students then share their solutions with the rest of the class, which become the basis for rich classroom discussions.

Learner-centered design. Yet merely creating software tools is not sufficient; they need to be integrated around the learner's needs and abilities (Quintana, Shin, Norris, & Soloway, 2006). Quintana et al. promote learner-centered design, which provides scaffolds that guide children's progression from novice to expert, as opposed to user-centered design, which assumes users already know how to do a task (e.g. write) and just need to learn a new digital tool (e.g. word processing). A word processing program is not intended to teach users how to write, but instead to teach how to use the tools to realize the user's vision of the desired output. User-centered design tries to make tools as easy and intuitive as possible. In contrast, learner-centered design assumes that the users need to learn both a new task and new tools. Learners have varying levels of expertise and need different scaffolds to learn new information. Digital tools designed for learners should provide scaffolding to guide the learners. The goal is not to make the tools easy, but to provide the appropriate level of challenge. For example, the authors describe *Symphony*, software built to support scientific inquiry among middle-school students. Because planning is an important part of the inquiry process that is often overlooked by novices, *Symphony* provides an explicit planning space with a process map and activity guides for reference. All students needed to complete a plan before continuing, but the amount of support they received in creating a plan differed based on their use of reference tools.

Influence of gaming. For young children it is important that learning activities be engaging. Too many computer games with an educational focus tend to fall flat with children, and the educational arena has much to learn from video games (Gee, 2005). Games that are popular with children generally involve fantasy, challenge, curiosity, and choice (Malone &

Lepper, 1987; Malone, 1981). Challenge is integrated into educational computer games by matching activities to a student's level, providing just the right scaffold and level of difficulty so that the activity is neither too dull nor too difficult. Ideally educational games should assess a student's understanding and adjust the game accordingly. Yet a game that focuses on adding might match a particular student's level but not inspire her to keep playing because it offers little more than drill. On the other hand another game might fascinate her and inspire curiosity because it shows the many ways to think about a given problem. Children should also be interactive participants in the game, making choices to affect the game as they play.

Increasingly game designers are striving to cultivate "flow" in gaming, the psychological state of being in the zone, theorized by Csíkszentmihályi (1991). Flow is the feeling of being lost in an experience, focused entirely on the task at hand and generally spending significant time in the flow state. Flow experiences are supported by clearly defined goals and rules as well as appropriate challenges and immediate feedback. Csikszentmihalyi first examined the nature of flow in dance, rock climbing, and chess, yet game designers have adopted his theory in the design of entertainment-based digital gaming. Educational game designers are also incorporating flow theory into existing theories of educational game design (Kiili, 2005). Kiili puts forward the experiential gaming model (see Figure 1) that explores how to best integrate flow into educational games, discussing the role of the zone of proximal development (Vygotsky, 1978) in providing appropriate challenges, which form the heart of the model. The model also features an ideation loop in which learners generate possible solutions and an experience loop, in which they try out the solutions. Elements of the game design including the problems, goals, feedback, usability, and control over the game all affect both learning and the probability of entering into a flow state.

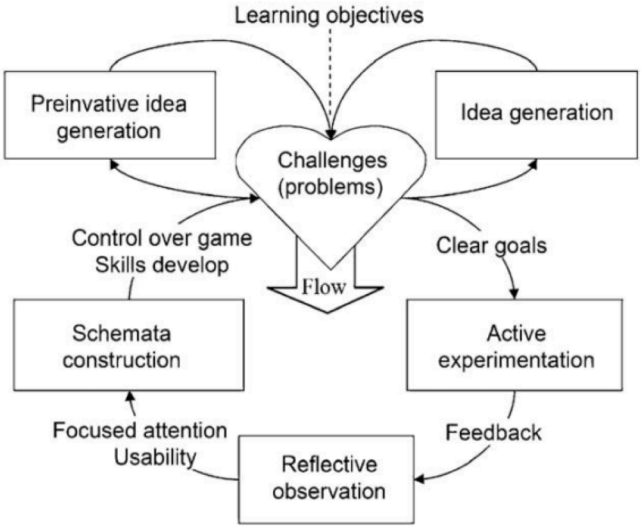

Figure 1
Experiential Gaming Model from Kiili, 2005

Pedagogical agents. Successful computer games often involve fantasy (Malone & Lepper, 1987; Malone, 1981). They tell a story that children can connect to. A successful fantasy does not need to involve sophisticated graphics; people's imaginations fill in the gaps (Reeves & Nass, 1999). The early Pokemon computer characters were fuzzy, pixilated shapes, yet children fell in love with the characters and the stories despite the simplistic graphics (Schell, 2010).

A strong element of any fantasy is the character. Learning games that incorporate a pedagogical agent (i.e. character that teaches) can show both improved learning and interest (Baylor & Kim, 2005; Baylor & Ryu, 2003; Gulz, 2004; Lester et al., 1997), especially when the agent provides interactive explanations (Moreno, Mayer, Spires, & Lester, 2001). Moreno et al. posited two hypotheses: a constructivist hypothesis, in which the digital character supports social learning, and an interference hypothesis, in which the character adds to the cognitive load and distracts from the learning. The researchers found that the digital characters do improve learning, supporting the social learning hypothesis. One way to increase the social nature of the fantasy is

by having several characters that interact with each other as well as the learner. Several studies have begun to explore the effects of multiple pedagogical agents on learning (Baylor & Chang, 2002; Baylor & Ebbers, 2003; Baylor, 2002; White, Shimoda, & Fredericksen, 2000). White, Shimoda, and Fredericksen assert that children understand the scientific process of inquiry better through multiple characters that each take on a different element of the process, yet they have not undertaken a systematic investigation of the role of multiple pedagogical agents in their software. Baylor and her colleagues have conducted several studies comparing a single agent to multiple agents with undergraduate participants, finding increased metacognition (Baylor, 2002) and increased learning under certain feedback conditions (Baylor & Chang, 2002). Baylor and Ebbers found that differentiating a pedagogical agent's dual roles of instructor and motivator into two characters increased learning and perceived helpfulness. Taken together, there is initial evidence that multiple pedagogical agents (versus a single agent) can improve learning. Yet more research is needed to determine how the multiple agents affect learning and what factors contribute to their effectiveness, including content area, population, and agent characteristics.

Development of Mathematical Thinking

Designing effective educational software relies on comprehensive understanding of the content domain, in this case: mathematics, more specifically small number addition strategies.

Strategy use. Children use a wide variety of strategies in problem solving, varying strategy choice from problem to problem and day to day, often using different strategies for the same problem on different trials within a single session (Siegler, 2007). Siegler asserts that strategy choice variability is an important area for research because variable strategy use is often beneficial and can predict improvements in learning, and recognizing variability gives a more accurate portrayal of cognitive development.

Addition strategies. In solving single-digit number combinations (NCs), children use a range of strategies (Carpenter & Moser, 1984; Cowan, 2003; Cross & Woods, 2009; Siegler, 1987). One of the simplest strategies is count-all. For example, solving 5 + 3, a student combines the two sets of objects and counts them all, beginning with 1. If he did not have objects to count, the child might hold up 5 fingers, hold up another 3, then count from 1. Some children using this strategy struggle with accuracy, especially with large numbers, and it is generally inefficient.

The next addition strategy children often use is count-on, in which the child counts from one addend. For example, solving 5 + 3, a child begins with 5, then counts on 3 more: 6, 7, 8. A child may count on with objects, fingers, or in her mind to achieve the answer. Some children using this strategy struggle with identifying a starting and stopping point, either counting 5, 6, 7 or continuing counting more than 3 times. There are two variations of counting-on: count-on-from-first (COF) and count-on-from-larger (COL), the later being a more efficient strategy (Carpenter & Moser, 1984). Solving 3 + 7, a child using COF would start with 3, then count-on 4, 5…10, whereas a child using COL would start with 7, then count-on 8, 9, 10.

As students practice single digit addition problems, many become automatic. Students can retrieve many of these NCs from memory, the most efficient strategy. Having some NCs memorized also aids in the derived facts strategy. Using this strategy, a child solves 6 + 7 by using his knowledge of 6 + 6. He understands that if 6 + 6 is 12 then 6 + 7 would be 1 more: 13. Children using this strategy often use either doubles (as above) or tens combinations to help them with more difficult NCs. To solve 8 + 5, a child might use the tens combination 8 + 2. She might think 8 + 5 = 8 + 2 + 3 = 10 + 3 = 13. With enough practice, children can become quite fast and efficient with derived facts (Baroody, 2003; Putnam, de Bettencourt, & Leinhardt, 1990)

Concept vs. procedure. It is vitally important that students learn not just the procedures of mathematics but also understand the concepts behind the procedures (Baroody, 2003; Fuson, Kalchman, & Bransford, 2005; Ginsburg, 1989). Baroody reviews the historical debate between the skills/procedural camp and the meaning/conceptual proponents, which eventually led to more modern views that both procedural and conceptual understanding are essential for true mathematical understanding. Baroody and Ginsburg (1996) promote a *schema-based* view of mathematical understanding that incorporates constructivist theory, social learning theory, and cognitive theory. This view posits that skills and concepts develop iteratively, with growth in conceptual understanding leading to greater procedural fluency, while becoming more proficient in a particular skill also leads to richer conceptual understanding. Research on the nature of conceptual and procedural knowledge acquisition supports this iterative hypothesis (Rittle-Johnson, Siegler & Alibali, 2001).

Procedural and conceptual knowledge influence interpretation of mathematical ideas. Gray and Tall (1994) argue that mathematical comprehension depends on flexibility in seeing notation as both processes and concepts, or *procepts*. For example, students may see 6 + 4 as a process, adding 4 to 6, or the concept, the sum of 10 being made up of two parts, 6 and 4. Once students become fluent with a process, it can become a concept and not require as much cognitive load, freeing up the ability to work with such *procepts* more flexibly. Derived facts are an example of this flexibility in action. In using a derived fact to solve 6 + 7, a student sees this number combination as more than just the process of adding 7 more to the number 6. The student may see the concept of 6 + 7 being related to the concept of 6 + 6 and understand the sums are also related. Gray and Tall find that more capable math students use number combinations strategies that require flexible *proceptual* understanding and posit that differences in

mathematics achievement may be due to a *proceptual divide*, with lower performing students seeing all NCs as separate processes that must be solved independently while more capable math students see NCs as procepts that can be used to help solve other problems.

Metacognition. Additionally, math software should promote metacognition, defined as "knowledge and cognition about cognitive phenomena" (Flavell, 1979, p. 1). Kuhn (2000) further distinguishes *metastrategic knowing* to specifically address one's understanding of the strategies available to solve a particular problem, which is integral to understanding how strategy use develops over time. Kuhn asserts that strategy selection takes place on a meta-level, and success or failure of a given strategy provides feedback to further develop meta-level understanding, which then influences future strategy selection, an iterative cycle that eventually leads to the increased adoption of more efficient and sophisticated strategies (see Figure 2). It is essential for interventions to address meta-strategic thinking in order to promote true strategy development.

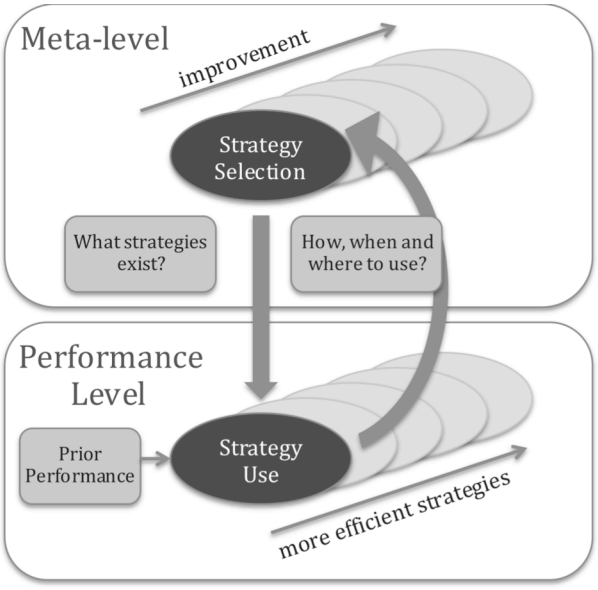

Figure 2
Meta-level Explanation of Strategy Development

In order to support meta-level processes, Fuson, Kalchman and Bransford (2005) recommend providing visual examples of student thinking and helping students communicate about their mathematical thinking, making the meta-level processes, which are typically internal, explicit and the subject of conversation. Encouraging self-explanation does not require a teacher-student conversation; self-explanation may be incorporated into mathematics software (Aleven & Koedinger, 2002). These researchers created two versions of their Cognitive Tutor Geometry™, one of which required students to select a justification for each problem step from a menu of options. Students using the self-explanation version were better able to explain their solutions and performed better on transfer tasks.

Differences in math performance. Typically, children transition from counting all to counting on in the early elementary grades, gradually adopting more sophisticated strategies including counting on, derived facts, and retrieval. The precise timing of the adoption of strategies varies considerably (for a comparison of studies see Table 1). These studies are far from an exhaustive list of the research in this area, but they represent a variety of methods and results.

Carpenter & Moser (1984) followed 88 mid-SES children from first to third grade, interviewing the children eight times over the three years using an interview format with researchers reading word problems aloud with small cubes available for children to use if they wanted. They found that at the beginning of first grade the predominant correct strategy was counting all (over 60%), while counting on with a correct answer accounted for only 14% of the trials. Counting on gradually increases during the first and second grade years, while by the middle of third grade almost all students had the problems memorized and used retrieval as their main strategy.

Table 1
Comparison of Select Studies Investigating Addition Strategies

Study	Secada, Fuson & Hall (1983)	Carpenter & Moser (1984)	Siegler (1987)	Geary et al. (2004)
Population	mixed-SES first-graders	mid-SES first-graders	high-SES first-graders	first-graders typical and with MD*
Measure	Cards with dots task	Verbal word problems	Written equations	Written equations
% Counting All	62%	~62%	1%	60% (MD) 20% (typical)
% Counting On	38%	14%	38%	27% (MD) 53% (typical)
% Derived Facts	n/a	n/a	9%	5% (MD) 17% (typical)
% Retrieval	n/a	~2%	44%	4% (MD) 8% (typical)

Note. For Secada et al (1983) these percentages are based on the percent of children who used counting on at least once (or not at all). For the other studies they reflect the percent of trials each strategy was used.
*MD refers to students with mathematical disabilities.

Siegler (1987) had quite different results with high-SES children and a task in which the problems were presented as equations with no manipulatives available. He found that first grade children were using retrieval as a strategy 44% of the time, the count on strategy 38% of the time, and derived facts 9% of the time, while they used count all only 1% of the time. These differences are likely attributable to either the higher SES status and/or the constraints of the task, principally, the lack of manipulatives and the presentation of formal equations versus verbal word problems.

Secada, Fuson, and Hall (1983) used a task specifically designed to encourage counting on, in which the researcher presents the child with two cards with a line of dots on each card along with two other cards that labeled how many dots there are (See Figure 3). During the first

three trials the researcher flips over the first card with dots so that only the label is visible and then asks the child how many dots there are. During the second three trials the researcher leaves both dot cards visible and again asks how many. The researchers found that 38% of the first graders in their study used counting on during at least one trial, while the other 62% only counted all.

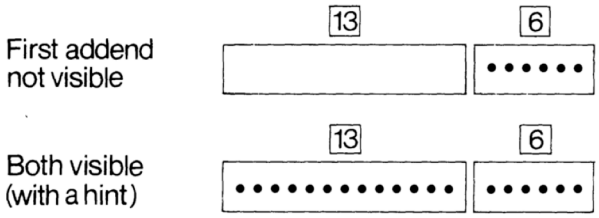

Figure 3
Count On Task from Secada, Fuson, and Hall (1983)

Children who have math learning difficulties (MD) have a delayed use of counting on, favoring the less sophisticated counting all strategy after their peers have begun counting on (Geary, Hoard, Byrd-Craven, & DeSoto, 2004). In this study first graders were presented with addition equations with no manipulatives. Students with MD counted all more than 60% of the time, compared to 20% of the typical students using count all. The first graders with MD counted on during 27% of the trials, while typical students counted on over 50% of the time.

In general young children from lower-SES backgrounds, such as those in our study, perform worse than middle-SES children on tests of mathematical performance (Jordan, Kaplan, Nabors Oláh, & Locuniak, 2006; Jordan & Levine, 2009). Also lower-SES children are 1.5 times more likely than middle-SES children to be diagnosed as having a math learning disability (Jordan & Levine, 2009). Yet their underlying conceptual understanding, measured by the types of strategies they utilize, has been shown to be equivalent to their middle-income peers

(Ginsburg & Pappas, 2004), although the timing of onset and percent of time each strategy is used may differ.

In summary, children use a variety of identifiable strategies to solve addition problems. These strategies follow a well-researched developmental trajectory. However strategy use differs greatly from student to student and task to task. Low-income students are more likely to have math learning difficulties and delayed strategy acquisition. While tasks that include countable objects may encourage more frequent counting all.

Behavior Detection

Another area of research that influenced the direction of the current study is the emerging field of behavior detection. Increasingly educational software designers are relying on data mining techniques to provide insight when interpreting computer log files of students' interactions with software. Mining these computer log files lets complicated patterns emerge and renders highly complex data interpretable, which in turn allows educators to design and build better educational software (Romero & Ventura, 2007). For example, a university may use data mining of online course interactions, including participation in discussions and forums, to determine that students who are at risk of failing the course participate in forums in particular ways. This information can be used to give the course professors an idea of which students are at risk or to let the designers of the course management software know how discussion features might be improved. Educational researchers use these techniques for a variety of purposes, including to create better models of student knowledge and affect, improve domain knowledge models, evaluate pedagogical techniques, and refine educational theory (Baker & Yacef, 2009).

Prediction. One important method in educational data mining is prediction, which may include regression, the predication of continuous variables, or classification, in which log data is

used to predict categorical labels. For example, a researcher may want to predict a student's score on a particular outcome measure based on his performance using educational software (continuous variable), or the researcher may want to predict success or failure on the next problem in the software (categorical). This method has been used to detect affect (Baker et al., 2012; D'Mello & Graesser, 2009), off-task behavior (Cetintas, Si, Xin, Hord, & Zhang, 2009), gaming the system (Baker, Corbett, Roll, & Koedinger, 2008), and exploration strategies, (Amershi & Conati, 2009).

For example, Baker, Corbett, Roll and Koedinger (2008) used data mining methods to predict whether students were gaming the Cognitive Tutor software in a way that was harmful to their learning. Researchers used peripheral vision to observe students using the Cognitive Tutor for 20-second intervals and code their behavior as gaming, off-task, on-task talking, or using the software. The researchers then assessed whether students improved from pretest to posttest, classifying previous instances of gaming as harmful when students did not improve. Using the log files of student actions, the researchers created a model to predict instances of harmful gaming.

Discovery with models. Increasingly, researchers are using data mining techniques in a discovery with models approach, wherein the data-mining model is used as one part of a larger analysis (Baker & Yacef, 2009). For example, Beck & Mostow (2008) use *learning decomposition* to discover the most effective type of reading practice for different sub-groups of students. Their goal was to figure out how to weight the different types of practice to best match the learning curve of individuals as well as groups of students. They found that for most students reading a variety of texts was more helpful than rereading a particular text, yet a small subgroup (3%) did benefit more from the rereading. In another experiment, Jeong and Biswas (2008) use

data mining techniques to classify student interactions with a teachable agent, Betty's Brain, into three general patterns of behavior. Once identified, the researchers were able to evaluate how students using different versions of the software differed in the frequency and length of these behaviors to determine the effectiveness of each software version.

MathemAntics

The literature previously discussed, including research on educational software, strategy variation, the development of mathematical thinking, and educational data mining informed the design of the current study and one particular software activity, Think Facts. The platform for this research and design is the MathemAntics (MA) project, a series of mathematics software activities currently being developed for children ages 3-8 (Ginsburg, Carpenter, & Labrecque, 2011). The software teaches children about numbers in a playful, antic way. Children help a farmer count and organize chicks and other farm animals, using a variety of tools that the children control. MA activities cover many topics including quantity, equivalence, addition, subtraction, place value, multiplication, division and negative numbers. In the next section, I will discuss how these various theoretical perspectives influenced my design of the Think Facts activity, including the digital tools, strategy characters, and logging system.

Digital tools. Learner-centered design encourages the use of digital tools to scaffold learners. Think Facts uses tools to support children's use of more sophisticated addition strategies: counting on, retrieval and derived facts. The activity promotes retrieval with timers and repeated practice, while also providing supportive tools to help students count on and see the connections between the facts they know and related problems. Within addition, the activity provides three tools: the count-on tool, the doubles tool and the tens tool. These tools present visual models that children can manipulate to reveal the underlying structure of number as in a

microworld. For example, with addition, addends can be switched (commutativity) or broken apart (decomposition) and added in a different order. Children can discover these properties by manipulating the objects through tools.

Using the count-on tool to solve 5 + 2, a thought cloud appears with a 5-box and 2-box. The child first selects the larger number, and the 5-box becomes a continuous whole while the 2-box splits into two 1-boxes, displaying the series notation 5, __, __ underneath (see Figure 4). A more proficient child may then type in her answer, while a student who needs more support may click on the 1-boxes to see them jump onto the stack and reveal more notation 5, 6, ?

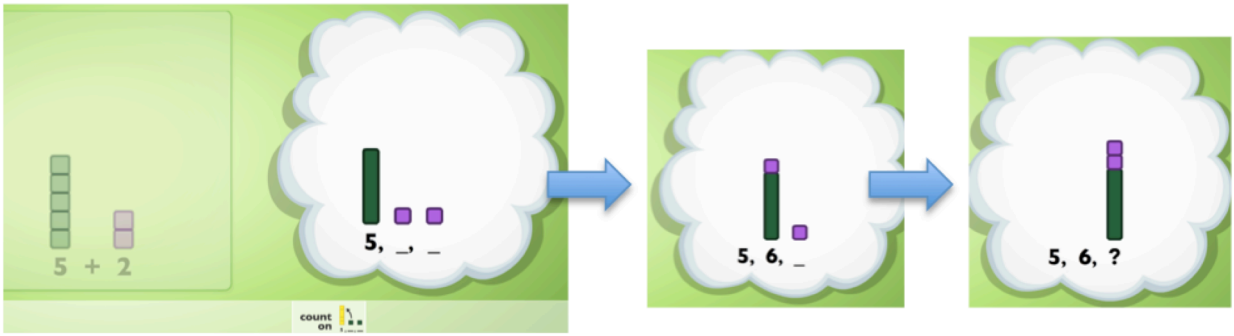

Figure 4
Count On Tool

With a problem such as 5 + 6, a child may press the doubles tool and a thought cloud appears with numbers represented as boxes. The cursor becomes a scissors tool, which he uses to break off the top block from the six, turning the problem into a familiar doubles fact, 5 + 5 with 1 extra (see Figure 5).

The same child could have also used the tens tool to help solve 5 + 6. In this case the thought bubble shows a transparent 10-box that he fills using the addend boxes. The one that is left over fall off as he puts the 5-box on top of the 6-box (see Figure 6).

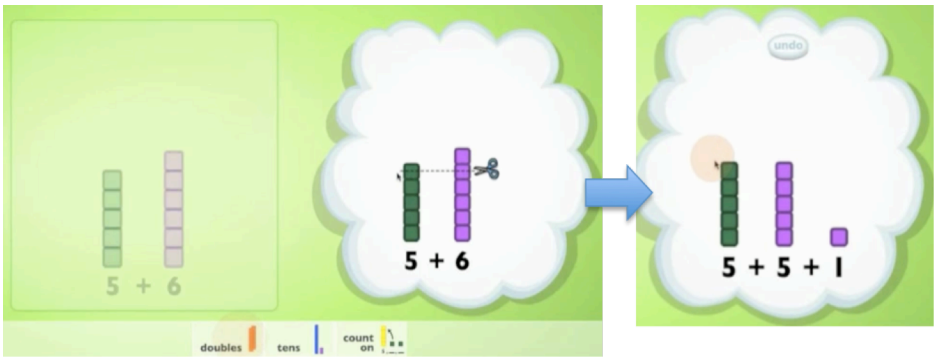

Figure 5
Doubles Tool

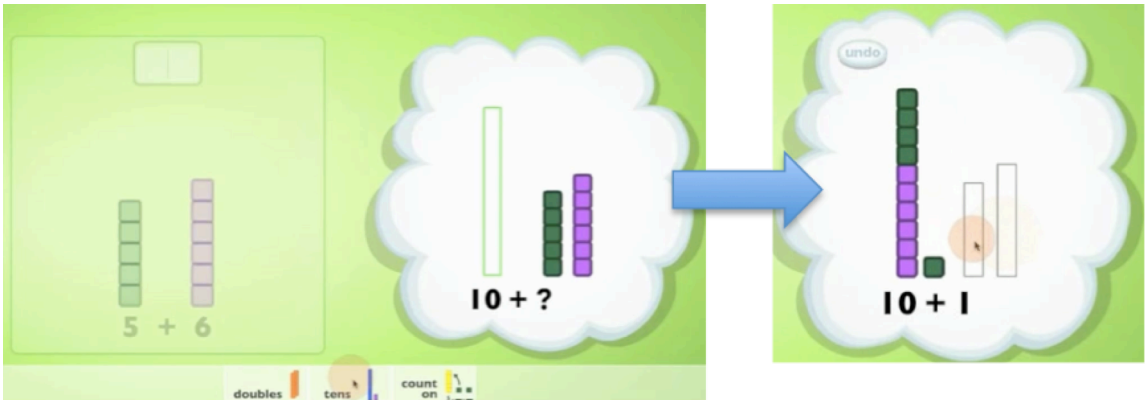

Figure 6
Tens Tool

These tools help visually explain how known facts can help solve more difficult combinations, making the connections between these concepts explicit. The three tools provide children with choices as they play, and support students with different amounts of scaffolding, which along with increasing problem difficulty helps keep the activity appropriately challenging. During the timed round, children have delayed access to the tools, encouraging them to think strategically without the tools immediately present.

As children play, they will also learn about three different addition strategies and get feedback about when each strategy is useful. For example, if they try to select the doubles tool

on a problem such as 5 + 2, they get the feedback, "There are no close doubles facts," and they must chose a different tool, which supports their metastrategic thinking by indicating in what situations a particular strategy is not useful.

Strategy characters. In order to engage students' understanding of narrative and make strategy use explicit, Think Facts introduces pedagogical agents that explain and model the different strategies. *Big Math for Little Kids*™ preschool curriculum includes the story *Henrietta Sees Numbers,* a book about a little girl who subitizes collections of objects rather than counting them. After reading the book, children talk about how they "see" numbers just like Henrietta. Reading about the character helps give the children language to describe their strategy. In the same way, I developed strategy characters to help children describe addition strategies, choosing to create four different agents to increase the social learning effects and embody the differences between each strategy.

Single-digit addition introduces four characters: Count-all Carl, Count-on Crystal, Memory Max and Favorite Facts Frieda. These characters serve as teachers, but their traits also manifest the differences between these strategies (see Figure 7). Count-all Carl and Count-on Crystal are a younger brother/older sister pair. Since counting-on is a more sophisticated strategy, Crystal is an older, wiser sibling helping her little brother find a shortcut. Children are familiar with this type of sibling dynamic and can relate their own experiences with older siblings, cousins, or friends to this virtual sibling pair.

Adults may assume that children want everything to be nice and sweet all the time and design games and activities devoid of any conflict. Actually, children crave tension and enjoy narratives that involve conflict and even villainy (Handler Miller, 2008). They do not want all the characters to be sugary sweet all the time. Crystal and Carl are not overly nice; they can

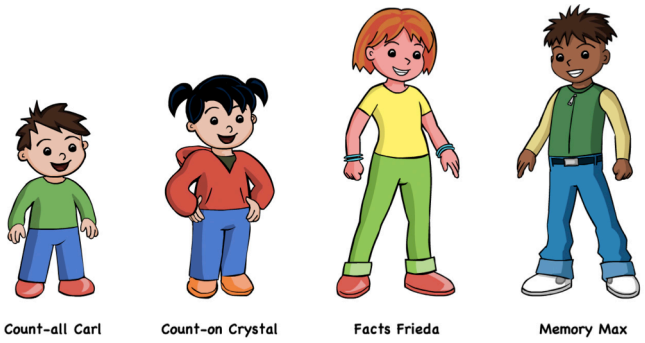

Figure 7
Strategy Characters

embody the tension siblings also experience as a way to encourage children's imaginations and fantasy. Crystal may be showing her brother a shortcut, but she is also showing off at the same time. Their tension is not only for the sake of the narrative, it also aids understanding. These two strategies are fundamentally distinct and children need to learn to abandon the count-all strategy, which feels safe and secure for a more efficient, abstract strategy. Just as real-life siblings may challenge us to grow and develop, so too can Crystal and Carl encourage children to try a new strategy.

Yet, Crystal, too, represents an immature strategy that should be abandoned for more efficient strategies. Memory Max and Favorite Facts Frieda are the end goal for efficient addition of single digit numbers. These two strategies can work together and complement each other, so Max and Frieda are not competitive siblings, but inseparable best friends. Visually, they appear even older than Crystal, embodying the mature character of these strategies. Max has a great memory for many number combinations, especially those involving doubles or tens combinations. Favorite Facts Frieda uses Max's facts to help her figure out harder combinations. Unlike Crystal and Carl who argue over strategies, these two complement one another and call on each other for help.

In order to increase narrative tension, MathemAntics also includes a villain, a neighbor farmer: Ortus, who likes to say "One Way – My Way." All four children are flexible in relying on each other to solve problems and readily ask for a back-up strategy. Even Crystal lets her brother Carl show how he solves a problem. Ortus, on the other hand, is an older farmer who is stuck in his ways, always using the same strategy.

Logging system. MathemAntics also includes a back-end logging system that records participants' actions during the game including mouse movements, clicks, and keys typed. The low-level log captures all actions with timestamps, while a separate high-level log captures predefined actions/timestamps. The high-level log file is organized with one problem per row and dozens of different variables, such as a numerical timestamp for beginning the trial, or a Boolean variable defining whether the child chose to use particular tool. We designed the high-level log for Think Facts to support data mining with some variables captured as children play and other variables calculated at the time of analysis. For example, the high level log captures speed for each trial as children play, while later, during analysis, I was able to calculate mean speed up to that point.

Current Study

The goal of the current study is to explore the effectiveness of math software in teaching single-digit addition strategies to first graders as well as detecting what strategies they use. Specifically, the study addresses the following research questions: 1) What are the effects of using the math software on student's accuracy and strategy development? and 2) When teaching about four different strategies, is it more effective to have four separate pedagogical agents or a single agent explain the content? 3) How well do students assess their own strategy usage? 4)

How well is the computer log able to detect what strategies students are using? 5) What features in the computer log are particularly important in the detection of students' observed strategy?

Methods

Design

The study utilized a pre/posttest, between-subjects microgenetic design with Condition (multiple v. single character) as the between-subjects factor. Both treatment conditions were highly similar, differing only on the one aspect of the software we were investigating: whether four pedagogical agents are more effective than one. Measuring children's addition strategy choice is complex because young children use a wide variety of strategies in problem solving, varying strategy choice from problem to problem and day to day. This makes the microgenetic method the most suitable to this kind of research (Kuhn, 1995; Siegler, 2007).

Participants

Participants were 47 first-grade students from three public elementary schools in an urban, low-income area. Participants were randomly assigned to one of two conditions: multiple characters or single character. To reduce uncontrolled variation, children were first grouped by pretest performance (low, med, & high), and then randomly assigned to each condition within block. The two treatment groups did not differ in terms of age, gender, or pretest scores.

Measures

The pre/posttest measures included the first grade mCLASS: Math™ Number Facts curriculum-based measure (CBM) and the Addition Diagnostic Interview (DI), measuring observed and expressed strategy usage (Lee, Pappas, Chiong, &Ginsburg, 2010). Both mCLASS measures were administered one on one by trained researchers using handheld devices. The handheld provided a script for the researcher to follow, space to enter students' responses, and, in

the case of the DI, follow up questions and strategies to code. The Number Facts CBM measured how many single-digit number combinations (NC) students could answer within one minute. The researcher said aloud the first NC and entered the student's response. If the student did not respond within three seconds, the researcher said aloud the next NC, unless the student was actively trying to work out the earlier problem. The Addition DI is a standardized clinical interview protocol measuring small number addition, order principle, zero principle, and mental math abilities. Researchers follow the script on the handheld device, coding observed, expressed, and backup strategies.

In addition, the posttest included a researcher-created measure of addition strategy awareness (ASA). The ASA contained four sub-parts: 1) strategy recognition, 2) definitional awareness, 3) demonstration, and 4) self-report. Strategy recognition was a computer-based task in which the child watched a short video of a girl solving a problem using a particular strategy (See Figure 8.1). The videos were scripted and rehearsed to ensure that the strategy was clearly displayed both in language and gesture. After watching each video twice, the child then identified which strategy the girl in a video used by clicking on buttons labeled: count all, count on, friendly facts, memory, or not sure. There were eight videos in all (each strategy twice), and the computer randomly assigned the order. The definitional awareness measure asked students to match a given definition to one of four (See Figure 8.2). The researcher read the strategy definition card aloud, asked the child to point to the strategy described, and then placed the definition card next to the strategy indicated. In the demonstration task, the researcher presented a sheet with an addition problem, strategy label, and character, saying, "Solve the problem the way ___ would by using ___" (See Figure 8.3). There were three problems presented this way: count all, count on and friendly facts, because memory is impossible to demonstrate if you do not

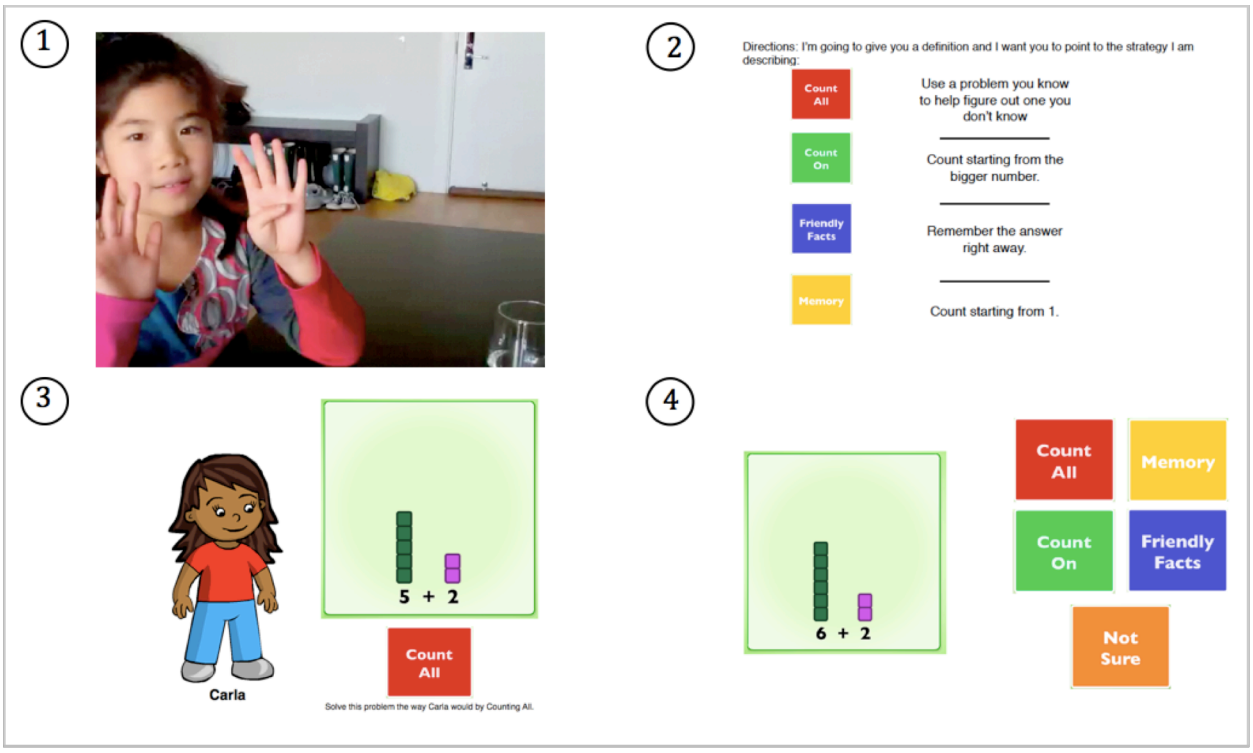

Figure 8
Images from Addition Strategy Awareness Subparts

have the problem memorized. In the multiple character condition, the child saw different characters on each page, while those in the single character condition were asked each time how the single character would solve the problem using the given strategy. The final self-report task involved first solving an addition problem that was presented on a card and then identifying the strategy used from a list (See Figure 8.4). The posttest included all four ASA subtests, whereas the pretest only included the strategy recognition subtest. During posttest the order of the tasks was counterbalanced as well as the order of problems within each task, using a Latin square design.

Using the software Silverback™, researchers simultaneously captured the actions on the screen and webcam footage of the student using the software. Prior to starting, the researchers established inter-rater reliability on the coding protocol for observed strategies, Kappa = .91 (see

Table 2). Cohen's Kappa compares two individual raters, so I instead used a multi-rater Kappa that accounts for situations in which the percentage for each code is not pre-determined (Randolph, 2005). Count All, Count On, and Derived Facts code strategies that are explicit and verbal, while Quick, Delay, and Count ? code either partial or complete internal use of strategy. A Quick answer, less than two seconds, may indicate the child is retrieving an answer from memory, guessing, or quickly using count on or derived facts. Count ? codes situations in which the child uses some counting gesture, such as head nodding or fingers, without clearly counting aloud, often mumbling or whispering. Delay codes when the strategy use is completely hidden, but the student does not answer quickly.

Table 2
Coding Protocol for Observed Strategies

Strategy	*Description*
Quick	fast response with no observed strategy
Delay	no observed strategy
Count All	audible counting starting from one
Count ?	partially audible or visible counting
Count On	Audible counting starting from either addend
Derived Facts	Use of a related fact

While using the software, the computer logged participants' accuracy in the task, self-reported strategy, tool usage, and response time. After compiling the local log files into a single spreadsheet, I added information about each trial's observed strategies from the researchers' recording sheets. Historical information, such as accuracy up to a particular point, was calculated using formulas.

Procedure

The study consisted of pre/post-testing and six intervention sessions of 10 – 15 minutes, divided into three parts: an interactive instructional video, an untimed tool round, and a timed round (see Figure 9). During each weekly intervention session, students worked one on one with one of five graduate students, trained to load the software settings, run the screen capture software, and code students' strategies. There were six levels of difficulty, which students progressed through as they achieved proficiency, defined as a maximum of one incorrect answer out of seven trials on the untimed round, (See Table 3). During week four, all students progressed to level three (if not yet achieved), so that all students were introduced to the four strategies and more than one tool.

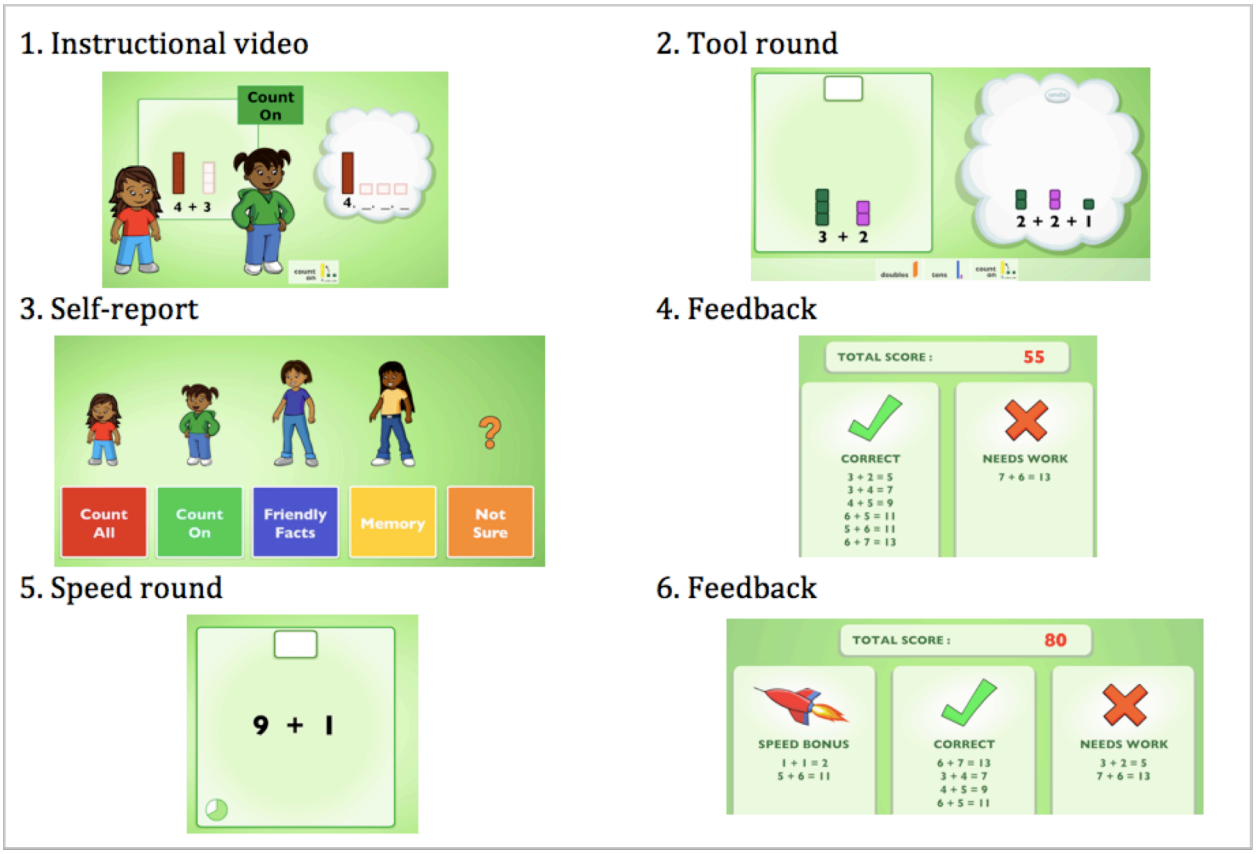

Figure 9
Images from Software for those in the Multiple Condition with Female Characters

Table 3
Characteristics of Addition Problems and Tools at Each Level

Level	Addends	Sum	Tools
1	one addend 1 - 3	sum < 10	count on
2	one addend 1-3	sum > 7	count on
3	near doubles	sum < 10	count on, doubles
4	near doubles	sum < 17	count on, doubles
5	any	8 < sum < 12	count on, doubles, tens
6	any	sum > 9	count on, doubles, tens

The instructional videos introduced the character(s) and modeled different strategies (See Figure 9.1). In order to mitigate the effects of gender preference and choice, researchers gave the child the choice of learning from a male or female character. Once chosen, the child saw either only male or only female characters for the videos (see Figure 10).

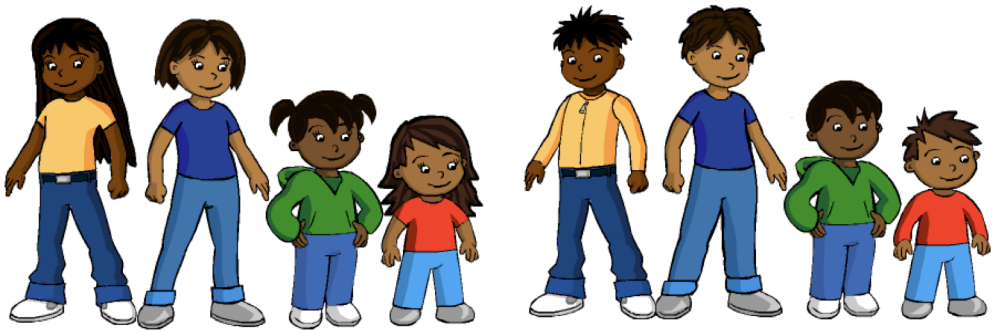

Figure 10
Set of Female and Set of Male Strategy Characters

During the first lesson video, the participants learn about counting all versus counting on. In the multiple character condition, the students meet two characters, whereas in the single character condition, one character explains the two different strategies. The scripts for these lessons contain the same content, adjusted only for the point of view. In subsequent videos the

students learn about the other strategies and the activity tools. Other videos ask children to decide which character should solve the problem or how the single character should solve it. Table 4 shows the lessons for the multiple character condition along with the software level and which students viewed each video each week.

Table 4
Instructional Videos with Corresponding Levels and Viewers

Video	*Level*	*Viewers*
1: Count-all and Count-on; introduce Count-on tool	1	All - week 1
2a: Who should solve this problem? (twice)	1 or 2	All - week 2
2b: Who should solve this problem? (twice)	1 or 2	Struggling - week 3
3: Friendly Facts and Memory; introduce Doubles tool	3	Proficient - week 3 Struggling - week 4
4a: Who should solve this problem? Who should be the backup?	3 or 4	Proficient - week 4 Struggling - week 5
4b: Who should solve this problem? Who should be the backup?	3 or 4	Struggling - week 6
5: Friendly Facts and Memory; introduce Tens tool	5	Proficient - week 5
6: Who should solve this problem? Who should be the back up?	5 or 6	Proficient - week 6

The second part is an untimed round in which participants practice using the tools to solve problems (Figure 9.2). After each problem, the self-report asks how the student figured it out and gives him strategy buttons to choose from (Figure 9.3). After the round ends, the students see a feedback screen reporting correct and incorrect response (Figure 9.4). The third part involves a timed round in which students see the equations for a limited time before the flash card turns over (Figure 9.5). The children can choose to flip the card back over, but they

only earn bonus points for those answered quickly. At the end of the timed round, they again receive feedback on those problems they got incorrect and correct, with a speed bonus for their quick responses (Figure 9.6)

Results

The results section follows the structure of the study questions, initially examining the effects of the software on students' accuracy and strategy development by first looking at the effects on all students, and in the second question, examining the effects of each condition: multiple characters versus a single character. Within the software, students also identified what strategy they used on a particular trial, so next I present an analysis of these self-reports. Finally, I examine the results of the two behavior detection questions: whether the software log is able to detect what strategies students are using and what features are salient in the detection models.

Software Effectiveness

Pre/posttest measures. There were three measures given at both pre and posttest: 1) mCLASS: Math Number Facts assessment measuring how many number facts a child can answer in one minute, 2) mCLASS: Math Addition Diagnostic Interview (DI), a structured interview coding accuracy, strategy use, and rule understanding, and 3) a researcher-created measure of strategy recognition (ASA-SR), in which students identify which strategy a girl in a video uses to solve the problem.

mCLASS typically reports a categorical score for the DI indicating advanced, rote, competent, and struggling. However, the interview captures and codes much more specific information, which I used to compute an Addition DI score (see Table 5). The interview begins with two small number addition problems that code accuracy, observed strategy, expressed strategy, and backup strategy. The next two problems focus on the zero principle, which codes

both accuracy and whether the student expresses the zero rule. The order principle problems code accuracy, observed strategy, and ability to express the order principle rule. Finally the mental math problems ask students to solve multi-digit problems in their mind, coding both accuracy and use of an advanced strategy. The Addition DI scores ranged from 0 - 21 points out of 21 possible points. The scoring rubric for the Addition DI demonstrated high internal consistency reliability on the pretest scores (Cronbach's alpha = .866).

Table 5
Scoring Rubric for mCLASS: Math Addition Diagnostic Interview

Component	*Category*	*Points*	*Description*
Small number problems (2)	Accuracy	0 - 2	2: exact accuracy 1: near accuracy (+ or - 1)
	Strategy	0 - 3	3: advanced strategy (count on or derived facts) 2: various strategies (e.g. chips and tallies) 1: single strategy or unsuccessful strategy use
Zero principle	Accuracy	0 - 2	1 point for each problem solved correctly
	Rule	0 - 1	1 point for expressing the zero rule
Order principle	Accuracy	0 - 2	1 point for each problem solved correctly
	Strategy	0 - 2	2: advanced strategy (count on or derived facts) 1: other strategy
	Rule	0 - 1	1 point for expressing the order principle
Mental math	Accuracy	0 - 2	1 point for each problem solved correctly
	Strategy	0 - 1	1 point for using an advanced strategy

The Number Facts scores ranged from 0 to 10 number facts correctly solved in one minute. The beginning of year benchmark for first grade is 6 number facts per minute and by the end of the year 11 facts per minute (mCLASS Guide, 2009). Students pretest score on the Number Facts assessment determined their ability block. A score of 0 or 1 indicated a low-

performer (n = 23), a score between 2 and 4 indicated a mid-performer (n = 12), and a score of five or higher indicated a high-performer (n = 12).

The Strategy Recognition task (ASA-SR) had a total of eight possible points with two videos of each strategy, count all, count on, friendly facts, and memory; students scores ranged from 0 - 7 points. Both the Number Facts and ASA-SR task distributions were positively skewed with more lower-scoring students, failing the Shapiro-Wilk test of normality. However, using Levene's test, the variances were equal across the two time points for both the Number Facts, $F(1, 85) = 2.09$, $p = .152$, and ASA-SR tasks, $F(1, 92) = 1.59$, $p = .210$.

Using a repeated measures ANOVA with two time points (pre and posttest), students improved on both the Number Facts, $F(1, 39) = 12.31$, $p = .001$, and Addition DI, $F(1, 46) = 32.02$, $p < .001$ (See Table 6). Complete data were available for all 47 participants on the Addition DI, but on the Number Facts task, seven data points were lost due to device error.

Table 6
Means Table for Number Facts and Addition DI

Task	Time	Mean	Std. Dev.
Number Facts (n = 40)	Pretest	2.40	2.70
	Posttest	3.78	3.26
Addition DI (n = 47)	Pretest	9.08	5.49
	Posttest	12.20	5.46

Students overall did not improve on the ASA-SR task, $F(1, 46) = 2.10$, $p = .10$, however adding ability blocks as a factor into the model revealed a Time*Ability interaction, $F(2, 44) = 3.40$, $p = .042$ (See Figure 11). Chance performance is two videos correctly identified out of eight opportunities, so at pretest many students scored near chance. Low-performers continued to

be near chance at posttest. Mid performers scored below chance at pretest and improved to slightly better than chance at posttest. Only the high performers (n = 12) progressed to a mean well above chance, t (11) = 2.74, p = .02.

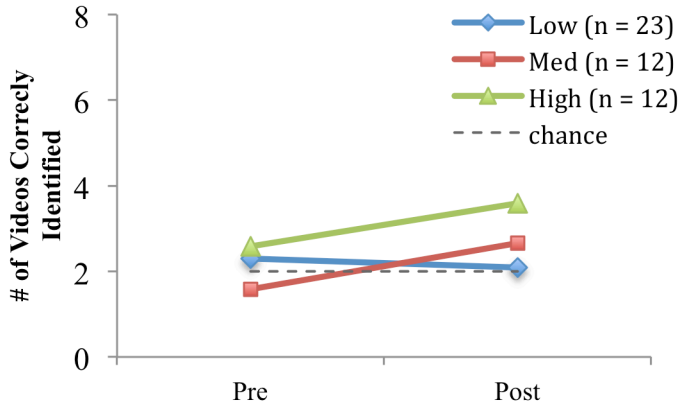

Figure 11
ASA-Strategy Recognition Task Scores at Pre/Posttest by Ability Block

Addition strategy awareness posttest. The strategy recognition task was part of a larger Addition Strategy Awareness measure with three additional subparts given at posttest only: definition, demonstration/application, and self-report. In the definitional task, students needed to match a given strategy definition to its name, four strategies for a maximum of four points. In general, students performed poorly on the definitional task with the median score = 1 being equivalent to chance performance and a mean = 1.57, SD = 1.31 (see Figure 12).

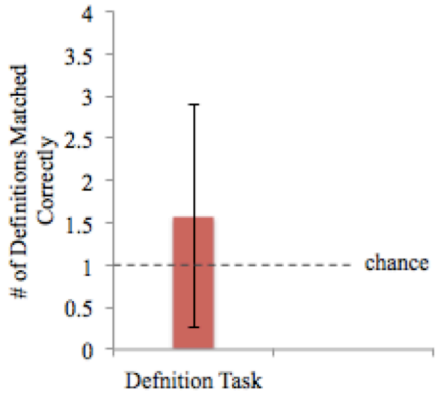

Figure 12
ASA-Definition Task Boxplot

In the demonstration/application task, the researcher asked the student to solve a given problem using a given strategy and then asked when it is good to use that particular strategy. There were three problems given: count all, count on, and friendly facts, because it is impossible to demonstrate memory if you do not have the problem memorized. Four points were possible for each problem, two for the demonstration phase and two for the application phase (see Table 7). Two researchers independently scored each trial using the scoring rubric and discussed trials on which there had been disagreement before coming to consensus on the appropriate score. There were no disagreements on the demonstration scoring because its criteria were highly explicit, however on the application task, the criteria were more subjective and there was initial disagreement between the two raters on 29 out of 141 trials. Out of six possible demonstration points, the mean score was 2.74 points (SD = 1.42), and out of six possible application points, the mean score was 2.38 points (SD = 1.41).

Table 7
Scoring Rubric for ASA-Demonstration/Application Task

Task	Points	Criteria
Demonstrate	2	Clear demonstration of the given strategy.
	1	Demonstration that indicates limited understanding.
	0	Demonstration conflicts with the given strategy or answer is off by more than 2.
Apply	2	Clear, logical explanation that highlights the strengths of a particular strategy (e.g. use friendly facts when you know an answer that is close or use count on to save time).
	1	General explanation (e.g. it's always good to count on).
	0	Don't know or non-mathematical explanation (e.g. when my mom says so).

Finally, in the self-report, researchers first asked students to solve a particular problem, identify the strategy they used from a paper that showed the strategy buttons from the software, and then explain how they used that strategy: “Show me how you counted all/ counted on/ used friendly facts, “ or “Tell me how you used your memory.” Again two researchers used a scoring rubric to independently evaluate the students’ responses (see Table 8). On the trials in which the researchers disagreed (22/141), they met to discuss the discrepancy and decide on a score. Out of 9 possible points, the mean score was 5.40 points (SD = 2.61).

Table 8
Scoring Rubric for Self-Report Task

Points	Criteria
3	Answer corresponds with the explained strategy and does not conflict with the observed strategy.
2	Answer corresponds with the explained strategy despite conflicting with the observed strategy, or answer corresponds with the observed strategy but does not correspond to the explained strategy.
1	Answer is considered plausible in the absence of a clear observed strategy or adequate explanation.
0	Answer conflicts with observed and explained strategies.

To evaluate if students ability grouping affects their strategy awareness, I used an ANOVA with the total ASA score as the dependent variable and ability block as a fixed factor. Since block was significant, $F(2, 44) = 8.94$, $p = .001$, a posthoc analysis using Tukey’s adjustment showed that high-performing students had higher strategy awareness scores than mid-performing students ($p = .022$) and low-performing students ($p < .001$). Looking more closely at each subtask reveals that in general the high-ability group had higher scores than the other two groups, but both the low and high-ability groups did well on the self-report task (See Figure 13). For that task, even if students are using a low-level strategy such as count all, they may still

receive all the points if they correctly identify solving it that way. The other tasks require students to have knowledge of counting on and friendly facts to get full points.

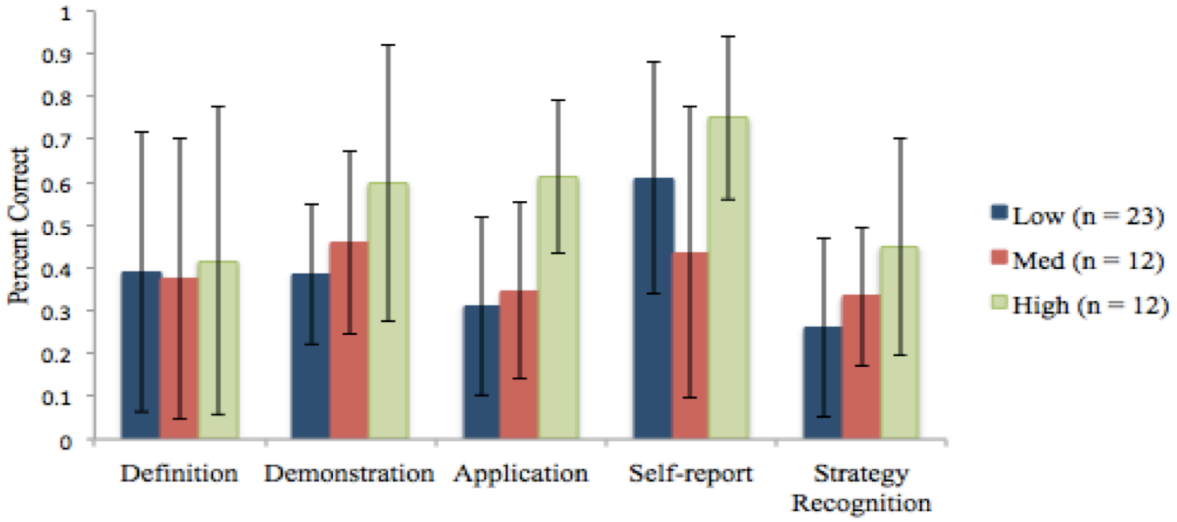

Figure 13
ASA Subtask Scores by Ability Block

In summary, on the pre and posttests, students improved on both a measure of fact fluency and holistic addition understanding. Mid and high-performing students improved in their strategy recognition, with high-performing students being the only group to perform better than chance. On the Addition Strategy Awareness posttest, high-performers outperformed the other groups on both the application and demonstration tasks. Students did best on the self-report task, especially the low and high-performers.

Session data. Analyzing the session data for accuracy and strategy-use presents some methodological challenges. Students progressed through the software at different rates depending on their accuracy during the tool round. Consequently, some students confronted easier problems than others. There were also two rounds with very different structure: the tool round (with three different tools) and timed round (with delayed access to tools), yet the dependent variables, accuracy and strategy, remained the same.

During the timed round, some students delayed answering until they had access to the tools and then relied on the tool to help get their answer, making the interactions similar to the tool round. The factor Tool categorizes what tool students used, but also has a value for the timed round if no tool is chosen. The four values of Tool are: no tool (timed round only), count on tool, doubles tool, and tens tool. If students used multiple tools, I took the tool that was introduced first, under the assumption that students would go back to a familiar tool for help if a new tool were not helpful.

To account for all of these factors, I used a Generalized Linear Mixed Model (GLM) using binary logistic regression with data organized at the trial level. The dependent variables were all binary (e.g. accurate/non-accurate or count all/other). The factors included Session, Tool, Question difficulty, and Ability Block with Student as a random factor. I ran each model with theoretically relevant interactions, removing any non-significant interactions iteratively. For any interactions that remained significant, I ran pairwise comparisons using a sequential Bonferoni adjustment. Bonferoni is a very conservative approach to post-hoc analyses, yet the statistical advantages of using the GLM to run analyses at the trial level outweigh the potential risks of using Bonferoni. The GLM model outputs estimated means for significant interactions and effects, which differ from a traditional mean in that they also account for the other factors and interactions in the model. In general, the following graphs use traditional means with standard deviation error bars. For cases in which the estimated means present a clearer picture of the model results, the axis labels indicate use of an estimated mean, and the error bars represent the estimated standard error.

Accuracy. In terms of accuracy I examined both exact accuracy and near accuracy (+ or -1), but both models showed the same significant effects, so I report the values for exact accuracy.

Looking at interactions, Block*Tool was significant, $F (6, 4221) = 6.66$, $p < .001$, with lower-performing students having worse accuracy in the timed round ($p < .001$) and mid-performing students having worse accuracy in the timed round than when using the count on or doubles tools. Not surprisingly, students were less accurate as problem difficulty increased, $F (6, 4221) = 34.84$, $p < .001$, but students did become more accurate across the sessions, $F (5, 4221) = 7.03$, $p < .001$.

Strategy. To examine strategy, I ran analyses for the individual strategies count all, count on, derived facts, and quick as well as a composite variable, advanced accuracy, defined as count on, derived facts, or quick. Refer to Table 9 for a general indication of strategy frequency. First, I examined students' use of Count All, finding a significant interaction for Block*Tool, $F (6, 4221) = 11.007$, $p < .001$ (See Figure 14). Looking at the pairwise comparisons reveals that low-performing students were less likely to count all when using the count on tool. Mid and high-performing students were less likely to count all during the timed round (no tool) or when using the count on tool. High-performing students were more likely to count all using the tens tool. These results reveal a potential design flaw in both the doubles and tens tools, which included discrete boxes that could be counted individually as opposed to the count on tool, which used a continuous box that discouraged counting all.

Table 9
Strategy Use Means (SD) by Ability

Ability	Count All	Count On	Derived Facts	Quick	Count ?	Delay
Low	.39 (.49)	.10 (.30)	.00 (.02)	.12 (.33)	.08 (.27)	.30 (.46)
Med	.23 (.42)	.12 (.33)	.01 (.11)	.27 (.45)	.09 (.29)	.28 (.45)
High	.10 (.30)	.20 (.40)	.01 (.10)	.32 (.47)	.12 (.32)	.26 (.44)
All	.27 (.45)	.13 (.34)	.01 (.08)	.21 (.41)	.09 (.29)	.29 (.45)

Note. Means represent a percentage of all trials in which the strategy was coded.
Low (n = 23), Med (n = 12), High (n = 12)

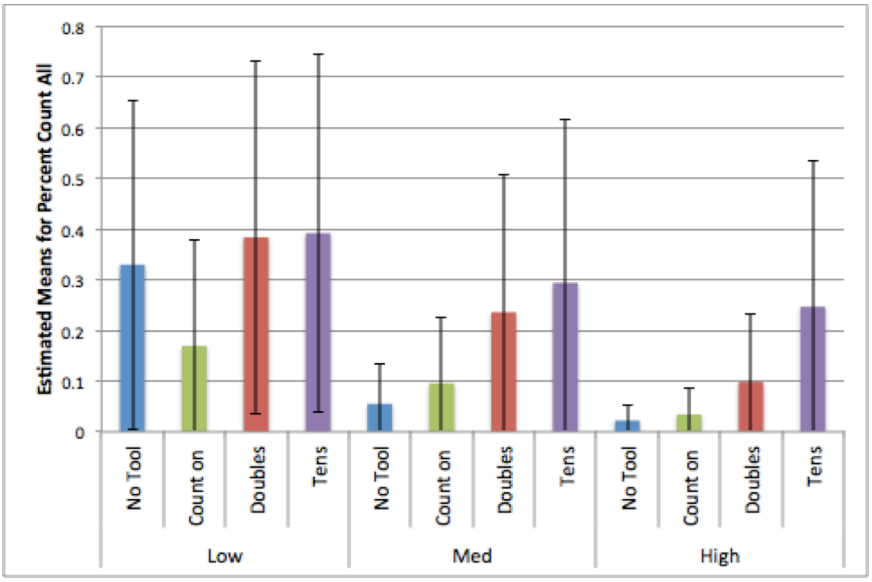

Figure 14
Use of Count All Strategy by Ability Block and Tool Used

For Count All, there were also significant effects of Session, $F(5, 4221) = 3.93$, $p = .001$, and question difficulty (QD), $F(6, 4221) = 15.77$, $p < .001$. Analyzing the pairwise comparisons for QD shows that students are less likely to count all to solve 1 + 1 ($p < .001$) and more likely to count all for problems such as 2 + 6 in which the sum < 10 and the second addend > 5 ($p = .024$). This effect may relate to students' preference for counting on from the first addend. When the first addend is smaller, the benefit of counting on is less and more students may decide to go ahead and count all. Examining the Session effect reveals that students are more likely to count all during Session 1 ($p = .049$), indicating that the software was encouraging some students to abandon counting all after the first session.

For Count On, there was a significant interaction for Block*Tool, $F(6, 4221) = 15.52$, $p < .001$. The pairwise comparisons reveal that low-performers were less likely to use Count On during the timed round ($p = .018$). Looking at QD, students are less likely to use Count On with the problem 1 + 1 ($p < .001$) and also with problems such as 2 + 6 in which the sum < 10 and the

second addend > 5 (p = .002). After Session 1, students were more likely to use Count On, F (5, 4221) = 8.21, p < .001 (See Figure 15).

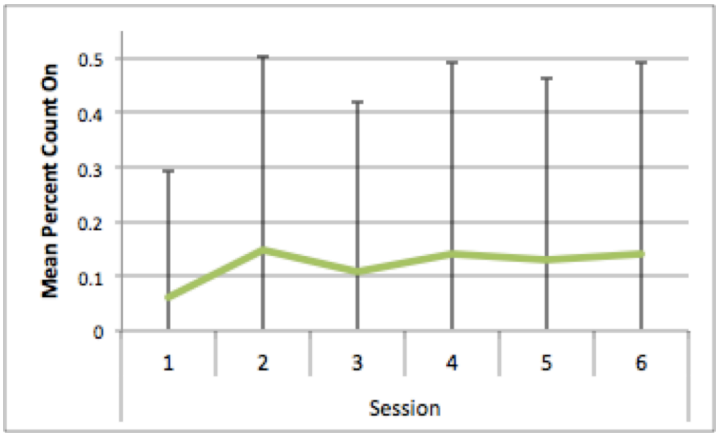

Figure 15
Use of Count On Strategy by Session

Derived facts were rarely observed during the intervention; in order to code this strategy, the child needed to verbally state how the given fact was related to another (e.g. "That's the same as 5 + 5 + 1"). When the non-significant interaction Block*Tool is included in the model, there is a significant main effect for Tool, F (3, 4221) = 2.73, p = .043 (See Figure 16). Pairwise comparisons show that students are more likely to use derived facts using the doubles tool than during the timed round (p = .045). This effect is no longer significant when Block*Tool is removed from the model.

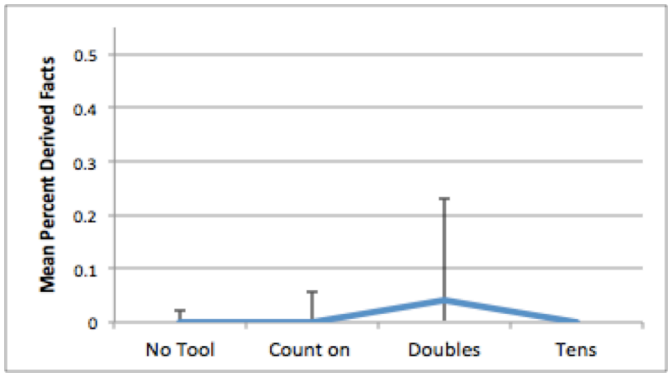

Figure 16
Use of Derived Facts Strategy by Tool Used

Students giving a Quick answer (less than two seconds) shows a significant interaction for Ability Block*QD, $F(12, 4215) = 2.57$, $p = .002$, with low-performing students being less likely to use the quick strategy for the problem 1 + 1 than either mid-performers ($p = .031$) or high-performers ($p = .008$). There is also a main effect for Tool, $F(3, 4215) = 125.39$, $p < .001$, with students being more likely to use the quick strategy when not using tools on the timed round ($p < .001$) (See Figure 17).

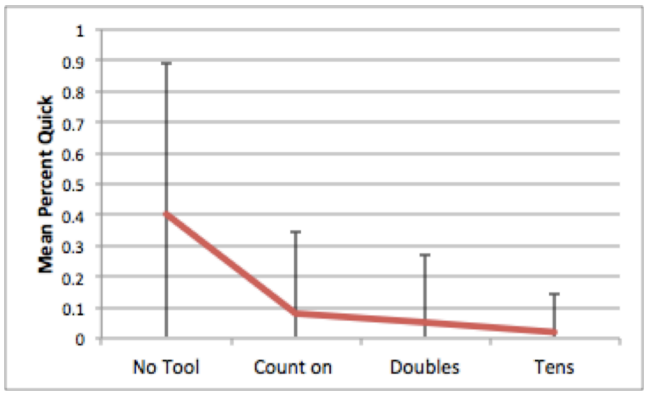

Figure 17
Use of a Quick Response by Tool Used

Students' use of advanced strategies in general, defined as Count On, Derived Facts, or Quick, has a significant interaction of Ability Block*Tool, $F(6, 4211) = 13.80$, $p < .001$ (see Figure 18). Pairwise comparisons show that for both the low-performers ($p < .001$) and high-performers ($p < .001$), use of advanced strategy is highest with timed round (no tool) > count on tool > doubles tool > tens tool. For mid-performers, the use of an advanced strategy is higher during the timed round (no tool) than with any other tool used ($p < .001$).

Use of an advanced strategy also has a significant interaction of Ability Block*Session, $F(10, 4221) = 3.05$, $p = .001$ (See Figure 19). Low-performers used more advanced strategies across the six sessions ($p < .001$). Mid-performers used more advanced strategies in Session 2

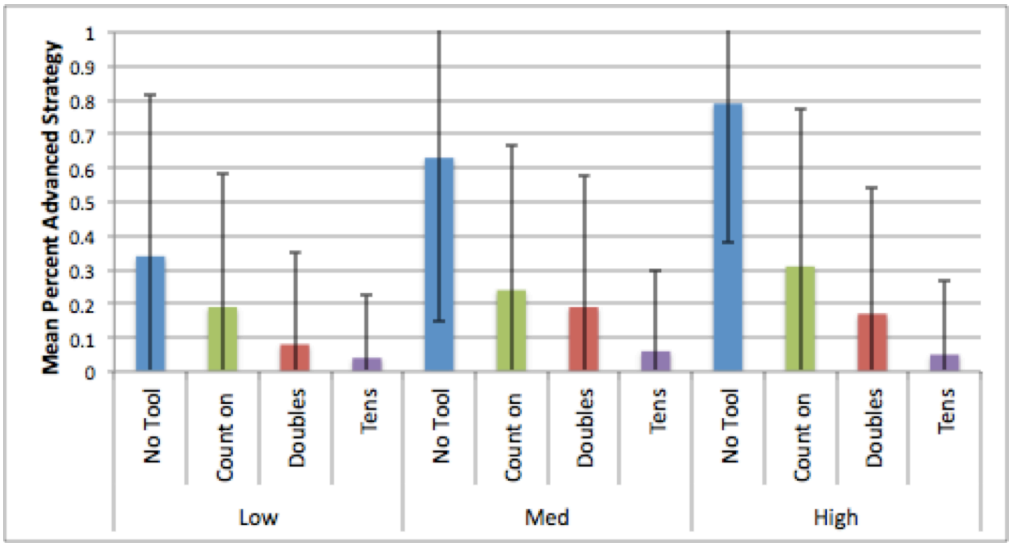

Figure 18
Use of an Advanced Strategy by Ability Block and Tool Used

than Session 1 (p = .004), but less in Session 6 than Session 2 (p = .027), indicating an initial rise followed by a decline. High-performers used advanced strategies equally across the sessions.

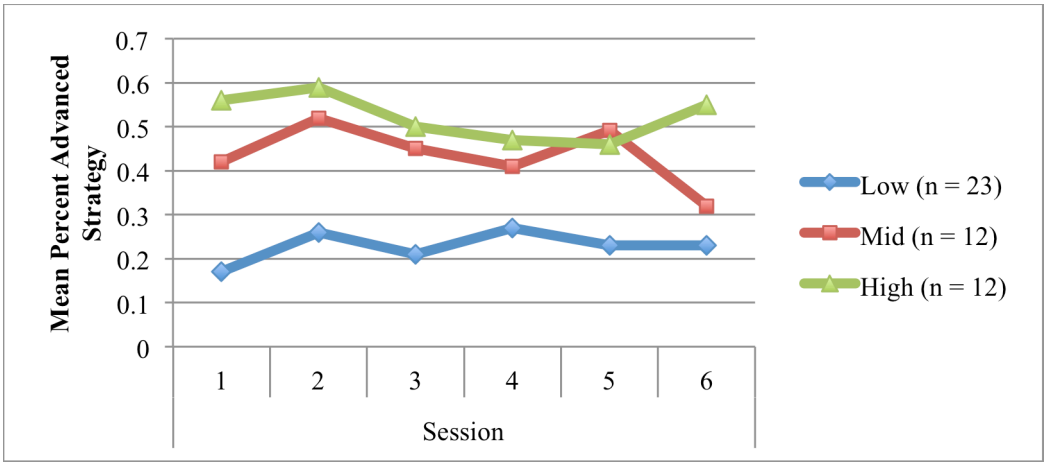

Figure 19
Use of an Advanced Strategy by Ability Block across Session

Advanced strategy (imputed). Since it is possible that students are hiding their strategy usage with an internal strategy, such as Count ? or Delay, I next built a data-mining strategy detector to impute these missing values. The model detecting Advanced strategy v. Non-advanced has a Kappa of .606 (A' = .833), indicating that it performs 61% better than chance. I

used this model to impute values for the Count ? and Delay trials, and then ran the new dependent variable, Imputed Advanced, through the Generalized Linear Mixed Model procedure. Again, there is an significant interaction of Ability Block*Tool, $F(6, 4221) = 16.976$, $p < .001$ (See Figure 20). After the imputation, the differences in the mid-performing group all become significant: timed round (no tool) > count on tool > doubles tool > tens tool ($p < .001$). However the imputation makes the differences between the doubles tool and tens tool for low and high-performers non-significant. There is also no longer a difference between the timed round (no tool) and count on tool for low-performers.

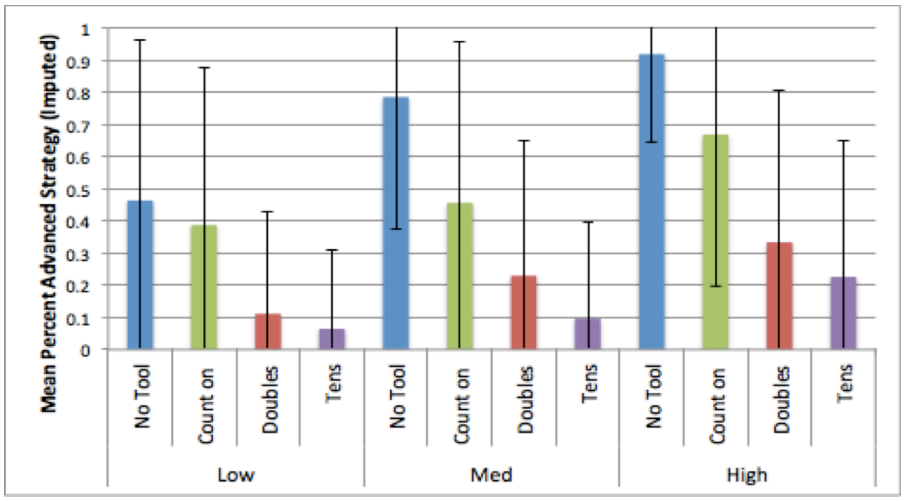

Figure 20
Use of an Advanced Strategy (Imputed) by Ability Block and Tool Used

In terms of what is happening across sessions, the imputation clarifies the effects. There is no longer a significant interaction, so the main effect of session becomes the story, $F(5, 4221) = 9.77$, $p < .001$ (See Figure 21). The pairwise comparisons show that students use more advanced strategies in all later sessions than in Session 1 ($p < .001$).

Advanced strategy + accuracy. Just using an advanced strategy does not guarantee that the student is really thinking, a student may answer quickly yet be wildly inaccurate. To evaluate

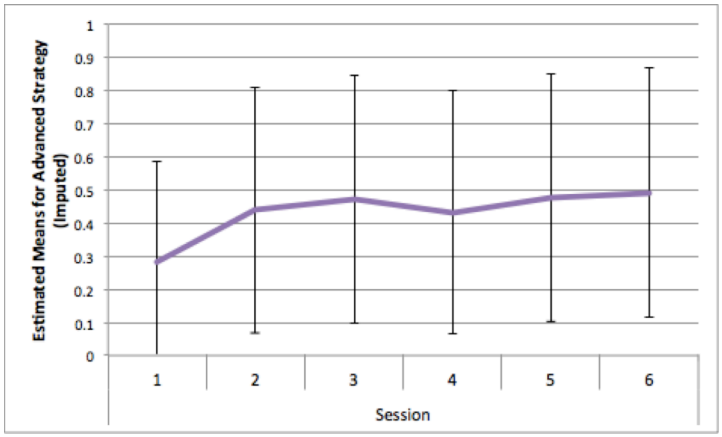

Figure 21
Use of an Advanced Strategy (Imputed) by Session

how successful students are in their use of advanced strategies, I created two dependent variables, one that combines the imputed advanced strategy with an exact answer and another that also includes a near answer. For the advanced strategy + exact answer, there was a significant interaction for Block*Session, $F(5, 4199) = 7.98$, $p < .001$ (See Figure 22). Low-performers improved across the sessions ($p < .001$), while mid-performers improved after

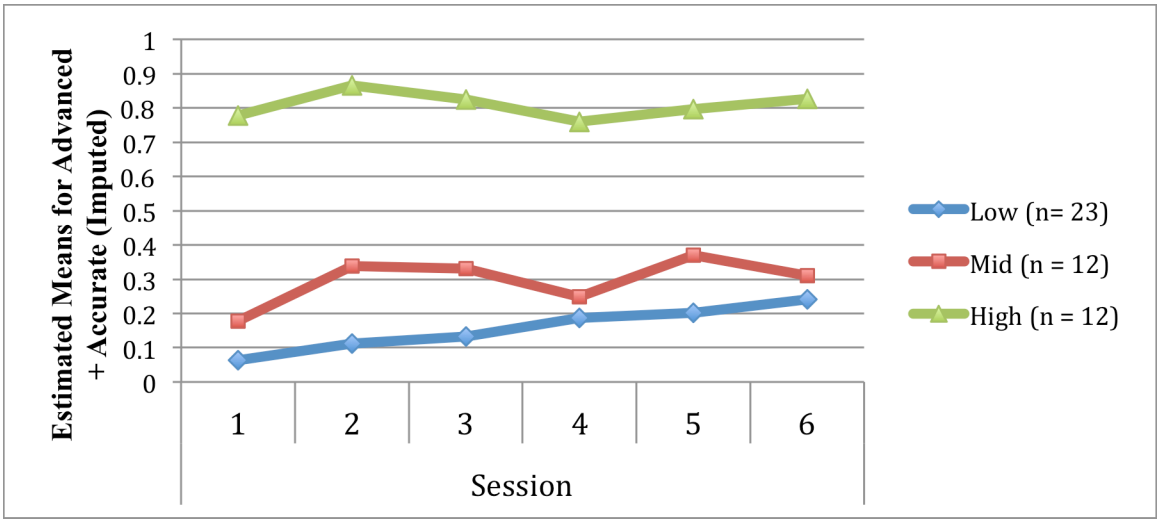

Figure 22
Accurate Use of an Advanced Strategy (Imputed) by Session and Ability

Session 1 (p = .001), and high-performers had consistently high use of advanced strategies with an exact answer. Advanced strategy + near answer had similar effects.

For the advanced strategy + exact answer, there was also a significant interaction of Ability Block*Tool, F (3, 4199) = 47.98, p < .001 (see Figure 23). Low-performers were most likely to use an advanced strategy correctly when using the count on tool and least likely when using the tens tool (p < .001). Mid-performers were equally likely to use an advanced strategy correctly on the timed round (no tools) and using the count-on tool, but were less likely with the doubles tool and even less likely with the tens tool (p < .001). High-performers were most likely to be successful with an advanced strategy on the timed round (no tools), followed by the count on tool, and least likely with either the doubles or tens tools (p < .001).

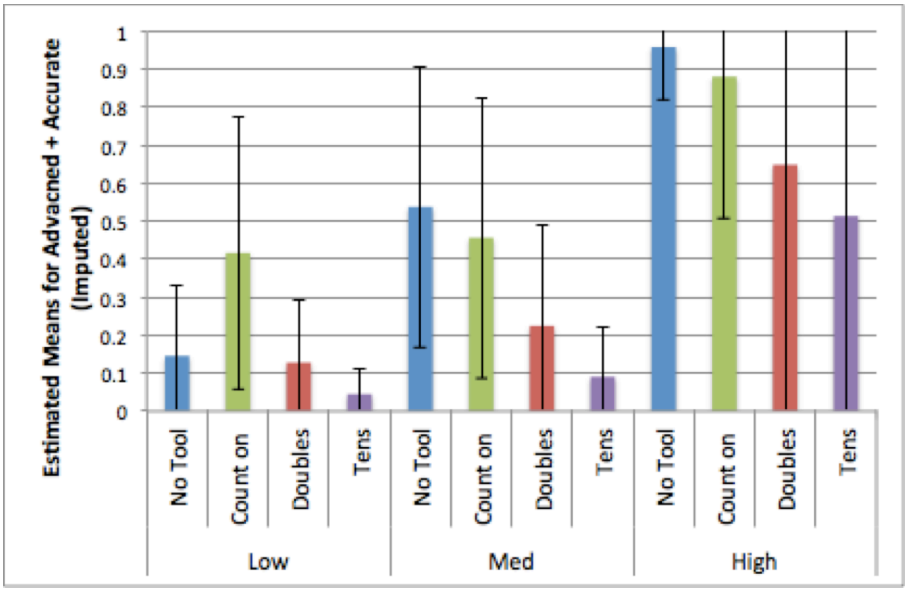

Figure 23
Accurate Use of an Advanced Strategy (Imputed) by Ability and Tool Used

For advanced strategy combined with a near answer, all pairwise comparisons for low-performers are significant: count on > no tool > doubles > tens (p < .001). For mid performers,

no tool > count on > doubles > tens ($p < .001$), while for high performers, no tool > count on > doubles = tens ($p < .001$).

In summary, students became more accurate over time. Students were also more likely to use an advanced strategy in later sessions than in Session 1. When combining accuracy and advanced strategy, low performers improved across the sessions, mid-performers improved after Session 1, and high-performers had consistent correct use of advanced strategies, even as problem difficulty increased. The count on tool promotes more advanced strategies than the other tools, and the timed round promotes advanced strategies for mid and high-performers.

Effects of Multiple Agents

Pre/posttest measures. The second question investigates the effects of multiple agents on students' accuracy and use of advanced strategies. First, I analyzed the pretest data to ensure that there were no differences between conditions on the three pretest measures. Using a MANOVA with Number Facts score, Addition DI score, and ASA Strategy Recognition (ASA-SR) score as three dependent variables and condition as a factor, there were no differences between the two conditions at pretest ($\lambda = .083$, $p = .969$). Using a repeated measures ANOVA for each pre/posttest measure with condition as a factor, there were no significant differences between groups. For the ASA measure (posttest only), using an ANCOVA with ASA total score as the dependent variable, condition as a factor, and pretest ASA-SR subscore as a covariate, there was not a significant difference between groups. In summary, contrary to my expectations, manipulating whether students saw a lesson from a single agent or multiple agents did not affect students' pre to posttest growth nor posttest performance.

Session data. To determine the effects of the multiple agents v. a single agent on students' performance within the software, I again used the GLM procedure using binary logistic

regression with data organized at the trial level with factors for: Student (random), Session, QD, Tool, Block, and Condition. I ran each model with theoretically relevant interactions, removing any non-significant interactions iteratively. For any interactions that remained significant, I ran pairwise comparisons using a sequential Bonferoni adjustment.

Accuracy. To evaluate whether students in one condition were more accurate, I examined both accuracy and near accuracy, finding no significant main effects or interactions involving condition.

Strategy. Looking at advanced strategy (defined as count on, derived facts, or quick), there was a significant interaction between Condition and Tool, $F(3, 4207) = 2.762$, $p = .041$. Students in the multiple character condition were more likely to be using an advanced strategy on the timed round with no tools ($p = .029$). However, after imputing a prediction of advanced v. non-advanced for the Count ? and Delay trials, this difference is no longer significant. Also, there are no significant interactions or main effects for Condition when looking at advanced strategy combined with near and exact answers.

Strategy Awareness

To answer my third question, I explored how well students assess their own strategy usage. For each tool round trial, students self-identified which strategy they used to complete the problem. For the first two to three sessions, students chose from three buttons: Count All, Count On, or Not Sure. After seeing the lesson on using friendly doubles facts, students chose from the complete set of five buttons: Count All, Count On, Friendly Facts, Memory, or Not Sure. Table 10 shows the frequency of each button choice.

Table 10
Percentage of Each Strategy Button Choice

	3 Buttons	5 Buttons
Count All	35%	16%
Count On	50%	21%
Friendly Facts	--	26%
Memory	--	28%
Not Sure	14%	9%

Researchers coded six strategies: Count All, Count On, Count ?, Delay, Derived Facts, and Quick. Each of the strategy buttons from the self-report strongly matched one of the researcher codes, Friendly Facts being equivalent to Derived Facts and Memory being equivalent to Quick. However, the strategy codes for Delay, Count ?, and Quick represent partial or complete internal strategy-use. These codes are each matched to self-reports that are plausible (Table 11). For example, with Delay a student displays no overt strategy, but may be internally counting all, counting on, or using derived facts, but is unlikely to be using memory because he/she took longer than two seconds to answer.

Table 11
Self-reports that are Plausible Matches to Observed Strategies

Student	Count All	Count On	Friendly Facts	Memory
Researcher	Delay, Count ?	Delay, Count ?, Quick	Delay, Count?, Quick	none

To evaluate how well students' self-reports match the researchers observed strategy codes, I used an inter-rater reliability Kappa with two raters: the student and the researcher. For each student, there are 42 observations, 7 tool round trials per session for 6 sessions. In order to differentiate strong matches from plausible matches, I calculated three separate Kappas for each

student: a conservative estimate, a middle-road estimate, and a liberal estimate. For the conservative estimate, ambiguous data are counted against the student (See Table 12), while for the middle-road estimate, all ambiguous data are removed (See Table 13), and for the liberal estimate, all ambiguous data are counted in the students' favor (See Table 14).

Table 12
Conservative Estimate for Kappa

	Count All	Count On	Friendly Facts	Memory	Not Sure
Count All	A				
Count On		A			
Derived Facts			A		
Quick				A	
Count ?					
Delay					

Notes. Greyed out cells counted as disagreement. Cells with "A" counted as agreement.

Table 13
Middle-road Estimate for Kappa

	Count All	Count On	Friendly Facts	Memory	
Count All	A				
Count On		A			
Derived Facts			A		
Quick				A	

Notes. Ambiguous cells removed from analysis.

Table 14
Liberal Estimate for Kappa

	Count All	Count On	Friendly Facts	Memory	Not Sure
Count All	A				
Count On		A			
Derived Facts			A		
Quick		A	A	A	
Count ?	A	A	A		
Delay	A	A	A		

After calculating three self-report scores for each student, I then averaged the three scores to compute the mean score for every student. Overall the self-report scores were quite low (see Figure 24). All of the distributions include negative values of Kappa, indicating a worse than chance agreement. Even the liberal estimate distribution has a mean less than 0.2, which is only slight agreement on Landis & Koch's 1977 scale.

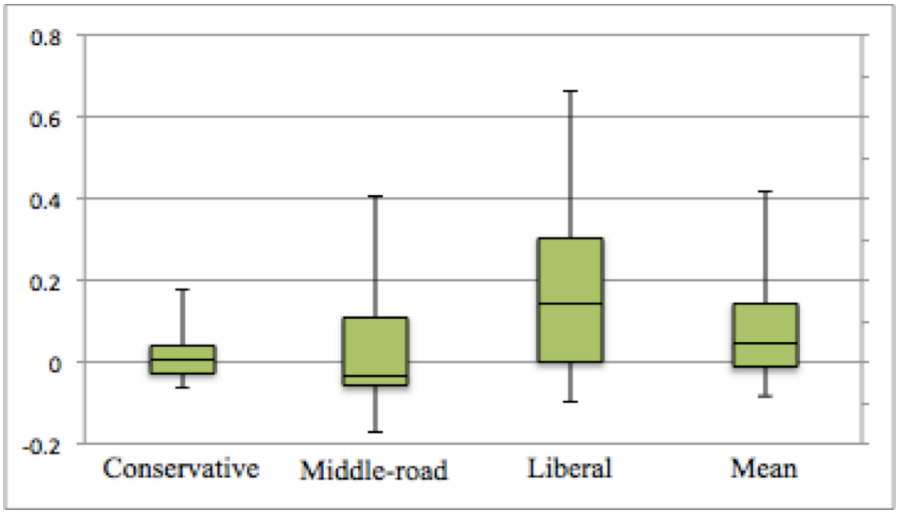

Figure 24
Self-report Scores: Conservative, Middle-road, Liberal, and Mean Estimates

To examine student characteristics that are associated with the self-report scores, I created correlation matrices between the four self-report scores and other student scores. Looking at the correlation of self-report to speed and accuracy reveals that children who work quickly in the software are more likely to give an accurate self-report, while those who both are accurate and use an advanced strategy have higher self-report scores (see Table 15). These correlations are stronger in the tool round, when students are giving the self-report.

Examining strategy use and self-report more closely reveals that students' use of Count All is negatively correlated with self-report, indicating that students who count all aloud are less

Table 15
Correlation Table of Self-report Scores with Speed and Accuracy

	Conservative	Middle-road	Liberal	Mean
Mean Time (Timed)	-0.04	-0.252	-.357*	-0.26
Mean Time (Tool)	-0.009	-0.157	-.430**	-.329*
Adv Strat + Accuracy (Timed)	-0.014	0.195	.335*	0.225
Adv Strat + Accuracy (Tool)	0.195	0.274	.393**	.327*
Mean Accuracy	0.25	0.194	0.15	0.169

Note. Adv Strat = Advanced Strategy (Quick, Count On, or Derived Facts)
** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).

accurate in their self-reports, indicating either lower meta-cognitive skills or the desire to appear more advanced (see Table 16). Students who have an internal use of strategy (Delay) have higher

Table 16
Correlation Table of Self-report Scores with Strategy-use

	Conservative	Middle-road	Liberal	Mean
Percent Count All	0.102	-0.268	-.612**	-.416**
Percent Count ?	-0.082	0.097	0.077	0.039
Percent Delay	-0.094	0.114	.474**	.339*
Percent Count On	0.091	0.201	0.204	0.176
Percent Derived	0.202	.313*	0.233	0.271
Percent Quick	-0.132	0.063	0.275	0.121
Percent Advanced	-0.027	0.203	.362*	0.228

** Correlation is significant at the 0.01 level (2-tailed).
* Correlation is significant at the 0.05 level (2-tailed).
Note. Strategy-use scores are the percentage of trials a strategy was utilized by the student.

liberal estimates of Kappa, which is likely due to the delay strategy counting as agreement with all strategy choices except Memory in the liberal estimate. Students who use Derived Facts have stronger self-reports as do students who use advanced strategies in general, defined as Count On, Derived Facts, and Quick. In order to code Derived Facts, a researcher needs to observe a student making a verbal connection between the problem they are solving and a related fact. Students who naturally verbalize their thinking in this way may be more adept at correctly identifying the strategy they just used.

Student characteristics such as age, gender, block, and condition revealed no significant correlations with the self-report scores. The self-reports also did not correlate to any of the posttest measures. Even the Self-report sub-task of the Addition Strategy Awareness (ASA-SR) measure did not correlate with the session self-report scores. The ASA-SR task is a pen and paper version of the software, showing students an equation with blocks for each number, similar to the presentation in the software. After solving the problem, the researcher asks the student to select what strategy they used from a paper that showed the same five button choices from the software, followed by a prompt for an explanation of how they used that strategy. The correlation between the ASA-SR and the Mean Self-report Kappa was only .129, indicating that students respond differently when the computer asks them what strategy they used compared to when a researcher asks.

In summary, students did not accurately report the strategies they used within the software. Students who progressed quickly through the software were more likely to correctly identify their strategy, and those who used advanced strategies had stronger self-reports. Students' ability to identify their strategy correctly in the software did not correlate with any of the Addition Strategy Awareness measures.

Detecting Strategy

To answer to my fourth question, I used data mining techniques to examine how well the features from the computer log detect researcher-coded observed strategies. The five trained researchers achieved a high level of inter-rater reliability on the strategy coding protocol (Kappa = .91), so in many cases the strategies coded closely approximate the true strategy each student used. Three of the strategy codes indicated a partial or complete internal use of strategy, and so the true strategy is unclear. For example, Count ? indicates some evidence of counting (such as pointing, head nodding, using fingers, or mumbling numbers), but the researcher was not able to clearly distinguish whether the student was using Count All or Count On. The strategy code Delay indicates a completely internal use of strategy in which the student took longer than two seconds to respond and did not demonstrate any overt strategy. A Quick answer meant the student responded within two seconds with no other strategy, and may represent a guess, a memorized answer or a quick use of either the derived facts or count on strategy.

The features that the computer log captured as students played included information about the problem, software settings, accuracy, tool usage, self-reported strategy, and speed (timestamps for all important actions). Using these original features, I calculated new features such as near accuracy, a response within + or – 1 of the actual answer, and trial speed subtracting the baseline speed of how quickly that same student answered the problem 1 + 1 in the same session, which is a problem most students had memorized. I also created historical features, taking into account prior performance. For example, I calculated a student's mean speed up to this point overall, within the same round, and for problems of similar difficulty. The complete feature spreadsheet contained 58 features (20 from the original data set).

In order to isolate effects, I created binary contrasts, for example, examining how well these features detected counting versus non-counting, in which the counting strategies (Count All, Count On, and Count ?) contrast against the non-counting (Quick and Derived Facts), dropping Delay, which may or may not involve counting. See Table 17 for a list of binary contrasts used.

Table 17
Observed Strategies Used in Each Binary Contrast

Contrast	Group 1	Group 2	Dropped
Count v. Non-count	Count All, Count On, Count ?	Quick, Derived Facts	Delay
Count All v. Others	Count All	All others	none
Count On v. Others	Count On	All others	none
Count All v. Count On	Count All	Count On	Quick, Derived Facts, Count ?, Delay
Internal v. Visible	Delay, Count ?	Count All, Count On, Derived Facts	Quick
Advanced-Accurate v. Non	Quick, Derived Facts, Count On	Count All	Delay, Count ?

Since Delay, Count ?, and Quick represent partially or completely internal strategy use, they are dropped from some of the contrasts. For example, with the Internal v. Visible, Quick is dropped because at its surface it indicates retrieval from memory, but it could also include students who use derived facts or counting on internally. Likewise the Advanced-Accurate contrast removes Delay and Count ? because these could either represent students counting all or using an advanced strategy. Removing ambiguous data ensures that all cases actually represent presence or absence of the tested strategy. However, removing these cases also runs the risk that these ambiguous data differ in substantive ways from explicit strategy-use, limiting the ability to use these models on new data. To test this limitation, the contrast Internal v. Visible tests

whether Delay and Count ? differ from the explicit strategies of Count All, Count On, and Derived Facts.

Educational data often contains a lot of noise, and it is important to choose a more conservative algorithm to avoid fitting to the noise in a particular data set rather than true behavioral indicators. The algorithm used in these analyses is the J48 Decision Tree (Quinlan, 1991), which is commonly used for educational data (Beck, Jia, Sison, & Mostow, 2003; Sao Pedro, Baker, & Gobert, 2012). The decision tree splits cases on a particular feature, for example, splitting accurate and non-accurate responses and then further splits the cases by other features, for example, splitting the accurate cases into those with a trial speed < 5 seconds and those with a trial speed > 5 seconds. At the termination of a particular branch in the tree, the algorithm states the outcome (e.g. counting v. non-counting) and how many cases from that branch were correctly and incorrectly identified. The J48 algorithm trims the best-fitting decision tree, eliminating any branches that do not add significantly to the model, which helps reduce the chances of overfitting. See Figure 25 for an example decision tree.

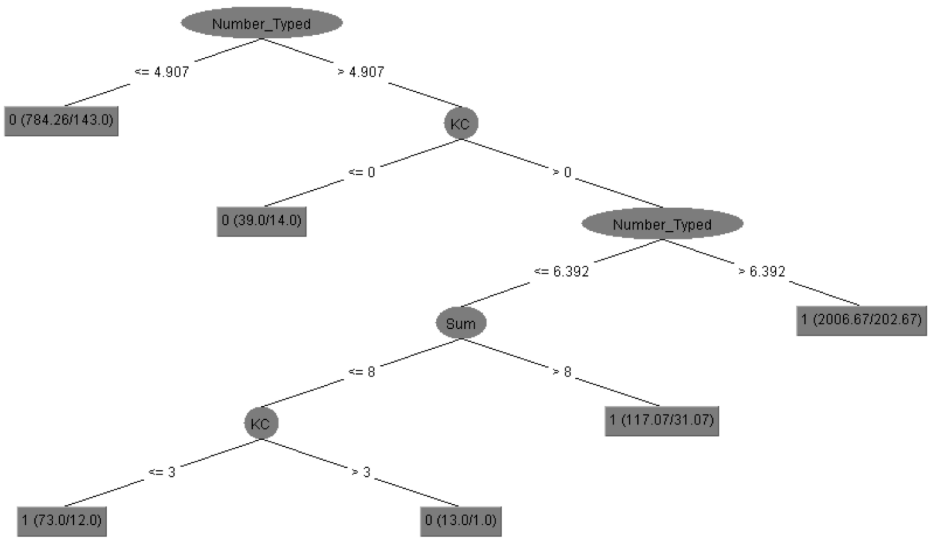

Figure 25
Example J48 Decision Tree

In order to evaluate the generalizability of a model, it is important to cross-validate using a training and testing protocol (Efron & Gong, 1983). I used a k-fold batch validation method, first randomly assigning the students to one of five batches and then repeatedly creating a model using data from four of the batches and testing the model's effectiveness on the withheld batch. Grouping at the student-level ensures that the resulting model will be effective with new groups of students, at least those who are similar to the original sample.

The cross-validation process reports a Kappa statistic on how well the model performs compared to chance. Kappa = 0 is chance, while Kappa = .37 indicates that the model performs 37% better than chance. Kappa is useful because it is easily interpreted and is familiar to the educational psychology community, however, this familiarity may also lead researchers to have overly high expectations. With human raters, researchers aim for a Kappa > .7 or .8, depending on the construct. However, computer-detected models are not expected to perform at these levels, especially for initial construct explorations, and Kappas < .4 are commonly published (Mavrikis, 2008). One important limitation of Kappa is that it does not take into account model confidence. For example, with the J48 decision tree, one particular branch may correctly identify 95/100 cases whereas another branch may correctly identify only 55/100 cases. Kappa treats the predicted values for these cases the same, which makes it less powerful and more conservative than other statistics.

One statistic for binary data that accounts for model confidence involves the Area Under the Receiver-Operative Curve (ROC). At a given probability threshold, there are four possible outcomes of a given model: true positive, false positive, true negative, and false negative (see Table 18). The ROC x-axis plots the ratio of false positives v. true negatives, and the y-axis plots the ratio of true positives v. false negatives. Using the example in Table 6, the data point

Table 18
Example Outcomes of a Count v. Non-count Model at the .5 Threshold

Outcome	Threshold	Confidence	Actual	# of Cases
True positive	0.5	> .5	count	100
False positive	0.5	> .5	non-count	15
True negative	0.5	< .5	non-count	200
False negative	0.5	< .5	count	5

corresponding to the .5 threshold would have an x-value = (15 false positives / 200 true negatives) and a y-value = (100 true positives / 5 false negatives), resulting in the data point (.075, 20). The curve is composed of data points from a variety of threshold levels. Chance performance is a 45-degree diagonal line. Models that perform worse than chance fall under this line, whereas models that perform better than chance fall above the line (see Figure 26). The goal in evaluating models is to optimize the Area Under the Curve (AUC). An AUC = .5 indicates chance performance and AUC = 1 is perfect. AUC values are generally higher than Kappa values for the same model. One limitation of the AUC statistic is that many software packages over-estimate AUC, especially in cases where multiple data points share the same confidence.

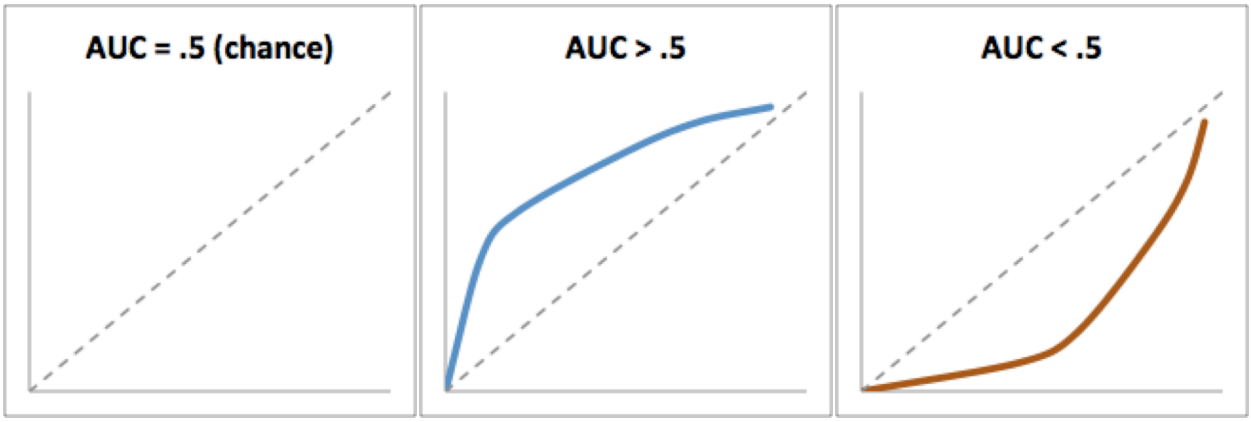

Figure 26
Example ROC Curve

One statistic that closely approximates AUC is A', which calculates the probability that when given an example of both count and non-count, the model will correctly identify which is the count. One important advantage of A' is that it is equivalent to the Wilcoxon statistic, which makes it possible to perform statistical tests comparing a model to chance or directly comparing two models (Hanley & McNeil, 1982). It is also possible to compute A' precisely using a relatively simple Java application; the following analyses use an application created by Baker and colleagues (www.columbia.edu/~rsb2162/misc.html).

In order to simplify the models and avoid the use of similar features in the same model, I used both a minimum correlation filter and forward selection. Starting each analysis with the full 58-feature dataset, I first trimmed the data to only include features that met a minimum correlation standard with the strategy contrast in question (absolute value > .08). Using Rapid Miner data mining software, I then created a single-feature J48 model using k-fold batch validation for each feature, removing any features in which Kappa <= 0 and AUC <= .5 from future models and selecting the feature with the highest Kappa. I paired the high-Kappa feature with each of the remaining features, testing each two-feature model, keeping the pair with the highest Kappa and removing any features that did not improve either Kappa or AUC over the single-feature values. I repeated this process until the model no longer improved. Once I had a final model, I exported the confidence for each case, and then computed A' using the Java application created by Baker and colleagues. The results of the strategy detection analyses are listed in Table 19.

Table 19
Strategy Detection Model Results

Contrast	Strategies	Dropped	Kappa	AUC	A'
Count v. Non-count	[CA, CO, C?] v. [Q, DF]	Del	0.672	0.841	.841
Count All v. Others	CA v. others	none	0.374	0.757	.744
Count On v. Others	CO v. others	none	0.128	0.615	.598
Count All v. Count On	CA v. CO	Q, DF, C?, Del	0.293	0.671	.644
Internal v. Visible	[Del, C?] v. [CA, CO, DF]	Q	0.292	0.665	.616
Advanced-Accurate v. Non	[Q, DF, CO] v. CA	Del, C?	0.673	0.899	.896

Notes. Abbreviations: Count All (CA), Count On (CO), Count ? (C?), Delay (Del), Derived Facts (DF), Quick (Q)

Contrasts that remove ambiguous data, not surprisingly have stronger models. Count v. Non-count and Advanced-Accurate v. Non, that remove cases where the strategy is unclear, each produce models that are 67% better than chance. However detecting an internal strategy is a much weaker model, only 29% better than chance, possibly because the behaviors categorized as Delay and Count ? share similar patterns of actions as overt counting. With the contrasts that include ambiguous data (e.g. Strategy v. Other), lower Kappas are expected because these contrasts include the Count ? and Delay codes, which likely contain students who are actually using the same strategy being tested but are categorized as Other. If the detector had a perfect Kappa = 1, it would actually be missing these students, so in these cases the ideal for Kappa is actually less than one. Trying to detect a single strategy from all others was moderately successful for counting all, 37% better than chance, likely due to this strategy being the least time efficient, and computer logs being particularly adept at capturing speed. However, trying to detect counting on from all others was the least effective model, only 13% better than chance, indicating that it shares patterns of behavior with at least one other strategy code. Trying to

differentiate counting all from counting on produced a model that performed 29% better than chance, which is an acceptable starting point, leaving plenty of room for improvements to both the software and detection models.

In summary, the software log is able to predict what strategies students are using, and these prediction models are generalizable to new students. Models are particularly strong when ambiguous data are removed from the analysis. Detecting counting all, which is the least time efficient strategy, is more effective than detecting counting on.

Computer Log Features

To answer my second question regarding the features in the computer log that are particularly important in the detection of students' observed strategy, I examined each of the decision tree outputs and the strength of each feature in the models. Count v. non-count produced the most straightforward decision tree (see Figure 27) with only three features: trial speed, sum, and question difficulty (QD). The model shows that students who responded very quickly (less than 4.9 seconds) were not counting, while those who took a long time (more than 6.4 seconds)

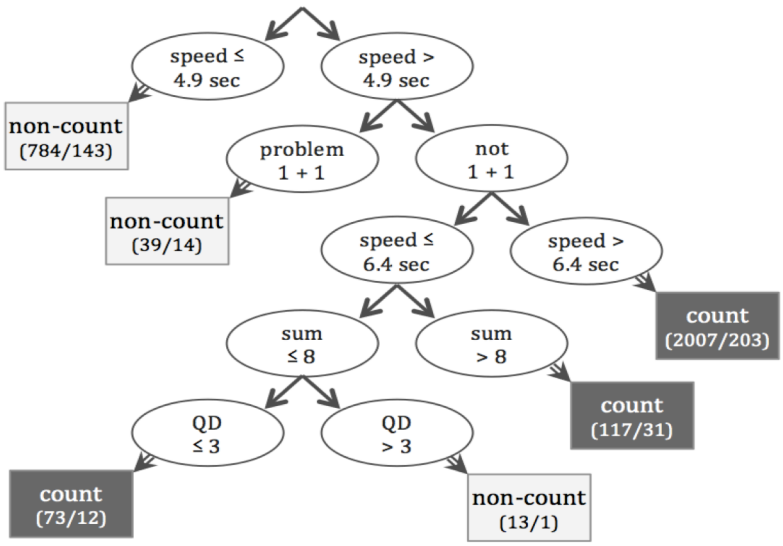

Figure 27
Decision Tree for Count v. Non-count

were counting, except if the problem were 1 + 1. For mid-length trial speeds, whether they counted depended on problem characteristics. If the sum were greater than 8, the child counted, but if the sum were less than 8, whether they counted depended on the QD, a measure of problem difficulty that takes into account both sum and addend characteristics. For QD <= 3, the second addend is always less than 5. There are only a few problems that have a sum less than 8 and a second addend greater than 5: 1 + 6, 2 + 6, and 1 + 7, and on these problems children who had mid-length trial speeds were not counting. Possibly the slightly slower trial speed was due to the time necessary to reverse the addend order or more time to find the answer 7 and 8 on the keyboard, since many children start scanning the keyboard numbers at 1. However, this particular rule only accounted for 14 cases, so its interpretation is limited.

The other decision trees were much more complicated, making it difficult to discern the overall logic. Instead, I examined each branch terminal and found those that accounted for the most cases, tracing those branches back to their origin to interpret select rules. For example, with Count All v. others, there are only four features: 1) trial speed, 2) average speed during the timed round up to this point (historical timed speed), 3) timed round speed for this same problem adjusted for 1 + 1 baseline speed (timed speed), 4) and the first tool selected, yet the resulting decision tree has 25 branch terminals, making the overall tree too complicated to interpret. One simple rule states that if the trial speed <= 5.4 seconds, then the student is not counting all, accounting for 968 cases (947 correctly identified / 21 incorrectly identified). Another states that if the trial speed > 5.4 seconds, but the historical timed speed < 5.1 seconds, then the student is not counting all (1313 / 181 cases). These two rules together basically state that if students are either very fast on this trial or generally very fast, then they are not likely counting all on this trial. One rule for counting all states that if the trial speed > 23.6 seconds and the historical timed

speed > 5.1 seconds and the first tool selected is the count-on tool, then the student is likely counting all (488 / 241). Because students first played for 2 – 3 sessions with only the count-on tool available, these cases are more likely to have been during the first few sessions rather than the later sessions. The confidence for this particular rule is not as strong as the others (67% v. 88% and 98%), however some of the cases that were incorrectly identified were Count ? and Delay, which easily could be internal uses of counting all.

Count On v. others was a particularly weak model, only 13% better than chance with all correlated variables in the model. Using forward selection to trim out features did not produce a stronger model, so I did not include this model in feature analysis.

Count All v. Count On included two features: historical speed on the same round and trial speed accounting for 1 + 1 baseline. If the historical speed <= 7.8 seconds, then the student counted on (272 / 43), but if the historical speed > 7.8 and the adjusted trial speed > 21 seconds, then the student counted all (642 / 105). These two rules together indicate that if students were generally fast, then they are likely counting on, but if the students are generally slow and very slow on this trial, then they are counting all. A third rule states that if the adjusted trial speed < 21 seconds, and the historical speed is between 10.3 and 25.7 seconds, then the students are counting all (526 / 120), which indicates that students who are generally pretty slow, but not extremely slow on this trial are still likely counting all.

The Internal v. Visible model had four features: 1) round (tool or timed), 2) near accuracy on either attempt, 3) time spent entering answer on tool round up to this point, which accounts for both historical speed and how long students had been using the software (historical time), and 4) use of the count-on scaffolding on this problem, which means that after selecting the count on tool, students clicked on the individual 1-boxes to see 5, __, __ turn into 5, 6, ? One internal v.

visible rule states that if students were near accurate on the timed round, then their strategy was visible (1278 / 405), indicating that students showed their strategy more often on the timed round, which did not initially display visual models. If students were near accurate on the tool round and their historical time < 415 seconds, then their strategy is internal (1261 / 483). Possibly as students are first introduced to the tool round, they are more likely to be thinking through the visual models internally or as problems become harder, they are more likely to visibly use a strategy. Another rules states that if students were near accurate on the tool round with a historical time > 478 seconds and no use of the count-on scaffold, then their strategy is visible (245 / 61). Since the count-on scaffold externalizes much of the strategy, visually showing how 5, __, __ becomes 5, 6, ?, the students who use this scaffold are less likely to display their strategy.

The final contrast examines Advanced-Accurate v. Non as a measure of more advanced thinking. In this contrast, in order to get credit for using an advanced strategy, the student also needed to be accurate. This model includes six features: trial speed, accuracy, round, first tool used, question difficulty, and the second addend. The first rule is obvious: if the answer is incorrect, then the strategy is not advanced-accurate (475). All other rules assume the answer is accurate. If students' trial speed <= 5.4 seconds, then the strategy is advanced (643 / 19), while if the trial speed > 22.8 seconds, then the strategy is non-advanced (666 / 152), which makes intuitive sense. If students respond quickly, they are using an advanced strategy, whereas if they respond slowly, they are using a non-advanced strategy. On the timed round, if the QD = 1 or 2 (both addends less than five) and the trial speed > 7.4 seconds, then the strategy is non-advanced (182 / 23), a rule that takes into account software, problem, and speed characteristics.

Examining all the features in the strategy models, I grouped similar features together, for example, collapsing both trial speed and trial speed that adjusts for 1 + 1 baseline. I also arranged the features according to general categories: student characteristics of time and accuracy, software characteristics of round and tools, and finally problem characteristics (see Table 20). The forward selection process establishes the rank order of the features in each model. Features that incorporate students' speed are at the top of most models, while problem characteristics (if present) are the last features, helping to refine models rather than define them. The internal v. visible contrast is the only model that does not have a direct speed component, because historical time incorporates both speed and overall time using the software. Advanced-accurate v. non has

Table 20
Features in Each Strategy Contrast Model

			Binary Contrast				
			Count v. Non	Count All v. Other	CA v. CO	Internal v. Visible	Adv-Acc v. Non
Student	Time	Speed on this trial	1	1	2		1
		Historical speed		2	1		
		Timed round speed on same problem		3			
		Historical time				3	
	Accuracy	Accuracy					2
		Near Accuracy				2	
Software	Settings	Round				1	3
	Tools	First tool		4			4
		Count On scaffolding				4	
Problem	Problem	Sum	3				
		KC	2				5
		Addend 2					6

Notes. Adv-Acc means advanced strategy + accurate response. CA = Count All. CO = Count On. The features for each contrast are rank ordered to reflect the forward selection process.

two features from each of the student, software, and problem categories, in that order of importance, while count all v. other and count all v. count on are concentrated on time characteristics.

Because students were clicking on what strategy they used within the software and the computer log tracked this choice, students' self-report choice was a possible feature in each strategy detection model. Interestingly, the students' self reports did not appear as a feature in any of the strategy detection models. The self-report did not achieve the minimum correlation benchmark (absolute value > .08) for the Count v. Non-count, Count All v. Other, and Count On v. Other contrasts. In the Internal v. Visible model, the self-report as a single-feature model had a Kappa = .07 and AUC = .542, but once Round was identified as the primary feature (Kappa = .19, AUC = .601), including self-report did not add to the model, indicating that it was actually acting as a proxy for Round, since there are only self-report values during the tool round.

In summary, time characteristics are particularly important in detecting strategy, while software and problem characteristics tend to refine the time characteristics to make stronger models.

Discussion

The discussion section will interpret the results of each research question, discuss implications for researchers, designers, and teachers, and finally, present the study limitations and areas for future research.

Software Effectiveness

Major findings. From pre to posttest, students increased their number fact fluency and also improved on a more holistic measure of addition understanding. Over the sessions, students became more accurate. Examining accuracy + advanced strategy indicates that low-performers

improved across the six sessions, while mid-performers improved after the first session, and high-performers consistently used advanced strategies correctly. Taken together, the session data indicates that low-performers, in particular, benefit from extended use of the software and more opportunities to practice using advanced strategies. Because the problems became harder each session, the consistent use of advanced-strategies by both mid-performers (after Session 1) and high-performers (all sessions) does not represent stagnancy, rather growth. These students continued to use advanced strategies despite the problems becoming more and more difficult.

In particular, the high-performers, who came into the intervention with advanced strategies, improved on the strategy recognition measure from pre to posttest. Likely these students were more ready for explicit strategy instruction because they had already adopted some advanced strategies. Given further use of the software, mid and low-performers, who adopted more advanced strategies during the intervention, might become better primed for the strategy instruction and also improve in strategy recognition.

Students' use of tools clearly influences what strategy they use. All students benefitted much more from the count on tool than either the doubles or tens tools. The count on tool was effective because it encouraged students to use the strategy it modeled. Part of its success likely stems from the use of continuous blocks rather than discrete boxes that may be counted individually. Both the doubles and tens tools inadvertently encouraged counting all because they displayed discrete boxes. Another part of the count on tool's success likely relates to counting on being a strategy that directly follows from counting all and being a strategy that all children eventually adopt. Young children are able to recognize counting on as a legitimate strategy before they begin using the strategy themselves (Siegler & Crowley, 1994).

However, teaching derived facts is inherently more complicated because many children never adopt this strategy (Gray, 1991; Putnam, de Bettencourt & Leinhardt, 1990). For example, Putnam, de Bettencourt, and Leinhardt interviewed third graders using puppets and three different scripts to elicit explanations of various derived facts strategies, but found that only 51% were able to give a complete description of the near doubles strategy in response to any of the three scripts and only 49% for the tens combination strategy. Given this, it is highly unlikely that students would progress from counting all as their main strategy to using derived facts over a short intervention. So it may be that these tools discouraged advanced strategies because of the use of discrete boxes or because they presented a strategy that the students were not ready for. The derived facts tools might have been more effective with older students or used only with students who had already demonstrated fluency with counting on.

Mid and high-performers also used more advanced strategies correctly on the timed round without tools. Low-performers were more likely to use an advanced strategy during the timed round, but were often inaccurate. Given the pressure to respond quickly, fewer students counted all. Mid and high-performers had internalized other strategies they could utilize correctly, but the lower-performing students tended to guess. To use an advanced strategy correctly, the low-performers needed the support of the count on tool.

Pedagogical Agents

Major findings. There were no strong effects of condition, multiple pedagogical agents versus a single agent, when other software features, such as tool-selected and question difficulty were included in the models. In terms of the pedagogical agents, the current study shows that both multiple and single agents can be equally effective in promoting adoption of more advanced

strategies. Possibly with more instruction, better design of the pedagogical agents, and more participants, differences would have been more apparent.

Metastrategic Awareness

Major findings. In general, the students are not very good at indicating what strategy they used by selecting a strategy from a series of buttons on the computer. Even giving the students the benefit of the doubt in each case where the strategy they selected is plausible, the Kappas are still quite low. Students who work quickly through the software are more likely to give an accurate self-report, and those students who use advanced strategies are more likely to give accurate self-reports. There may be several reasons that students who use advanced strategies have more accurate self-reports. The lessons presented some of the strategies as being better than others, so all students may have wanted to self-report the advanced strategies, with only those who actually used these advanced strategies reporting accurately. Alternately, students who use advanced strategies likely have better awareness of the strategy vocabulary and, hence, report more accurately. In general, the poor self-report results, taken together with the behavior detection results, clearly show that students' actions reveal more about the strategies they are using than their self-reports.

The posttest included a self-report measure that was similar to the software, differing in that the child indicated to the researcher what strategy they used by pointing to a strategy button icon rather than clicking a button on the computer. The researchers also asked a follow-up prompt, such as "Show me how you counted on," after the students made their selection. Calculating Kappa for these responses is not feasible because there were only three problems, so the responses were scored according to how well the explained and observed strategies matched the self-report. Compared to the other Addition Strategy Awareness (ASA) tasks, students did

much better on this measure, especially the low and high-performers. Students had the opportunity to receive full points on this measure even if they were using non-advanced strategies as long as their explained strategy matched the self-report and did not conflict with the observed strategy. Low-performers could use count all for all three questions and receive full credit so long as they also selected count all and explained counting all.

Students did not do very well on the other ASA measures. Unlike the self-report, these measures required knowledge of counting on and friendly facts to get full credit. The strategy recognition task had the lowest scores despite being a multiple-choice format. This task was administered on the computer with students selecting their response by clicking on a button, possibly indicating that these first-graders are particularly ill suited to responding to metacognitive questions on the computer as opposed to telling a researcher. Despite the low overall scores, mid and high-performing students did improve on the strategy recognition task from pretest to posttest, although only high-performers performed better than chance.

Children's weak self-reports in the current study contradict previous research in which children have been asked to describe the strategy they used. Siegler (1987) reported that students generally responded accurately to the prompt, "How did you figure out the answer to that problem," followed up by, "What number did you begin your counting on," when clarification was necessary. Siegler notes, "The videotaped records proved to be a useful supplement to the children's explanations in cases where the explanations were unclear and in the few instances where the children's overt behavior contradicted their descriptions of what they had done" (p. 255). By contrast, in the current study, within the posttest self-report, one-third of the responses either completely (27%) or partially (6%) contradicted the self-report. Geary, Hoard, Beard-Craven, and DeSoto (2004) report that students' general description of their strategy (e.g. verbal

counting rather than specifically counting on) agreed with the experimenter in more than 95% of trials.

A few important differences may contribute to the findings of the current study differing from Siegler and Geary et al's results. Neither of the other studies asked students to identify a particular strategy from a list. We had expected that asking students to choose their strategy from a given list would be easier because it involves recognition rather than recall, but the multiple choice nature may have instead led to more haphazard responses (e.g. pointing randomly to any strategy). The list also included strategy vocabulary that may have been unfamiliar to students. Each of the strategies was introduced and modeled in instructional videos, yet students may not have fully understood the new vocabulary from these videos, as evidenced by the low definition task scores on the ASA.

Neither Geary nor Siegler expected children to produce strategy-specific vocabulary. For example, in Geary et al's study, the students could simply state that they counted, while in Siegler's study if they replied counting, they were asked the follow up question asking what number they started with to further distinguish counting all from counting on. The software in the current study did ask follow up questions when students chose counting on, "What number did you start with?" or using friendly facts, "What fact did you use?" Students then selected from a series of choices that included a ? if they were unsure. In the posttest self-report, the researcher followed up by asking the child to show how they used a particular strategy. The current study also presented lessons in which certain strategies were presented as better than others (e.g. counting on being introduced as a "shortcut"), which may lead students to choose a more advanced strategy than they actually used.

The participants also differed in terms of socio-economic status: Siegler's participants were largely high SES, Geary's were largely middle SES, while this study included mostly low SES students. Finally, the participants differed in their strategy use. In Siegler's study, only 1% of trials involved students counting all, while in Geary's study, 20% of trials involved counting all (for typical students), and in the current study 27% of software trials involved counting all.

Many factors could have contributed to the poor self-report ability, but the findings of the current study indicate that interviewing children about their strategy usage is a more effective measure of strategy awareness than computer-based tasks. The temptation to click through the buttons quickly to get through to the next question may be an impulse too difficult to control for many young children. Asking open-ended questions, as opposed to multiple-choice questions, also seems to elicit more accurate self-assessment.

Strategy Detection

Major findings. The use of computer models in the addition strategy literature is not new. Shrager and Siegler (1998) describe the Strategy Choice And Discovery Simulation (SCADS), a computer simulation that models the process by which students choose and discover new addition strategies. SCADS includes problem characteristics, speed, and accuracy as well as working memory constraints for each strategy and a metacognitive system that evaluates performance and attempts to identify possible improvements. The goal of this kind of simulation is to determine what features create accurate models of development and identify what is happening inside the black box of strategy development. Since the SCADS model outperformed earlier simulations that did not incorporate working memory constraints or a metacognitive system, the researchers posit that these features are likely part of the developmental process.

Behavior detection uses computer-generated models in a different way. Instead of building a model to simulate what is happening inside the black box, behavior detection analyzes everything that is happening outside of the box, for example, comparing all student actions in the software to observed actions, in order to discover what a student is likely doing and/or thinking at a particular time. The goal is to predict what is happening to one particular student as opposed to making generalizations about development in general. However, as these strategy detection models improve and are tested on more and more students, it is possible that the features they identify will cast light on particular aspects of the developmental process. For example, they may eventually help to identify student characteristics that distinguish different developmental trajectories.

The current software is best able to detect contrasts that first remove ambiguous data. For example, the contrast Accurate + Advanced Strategy v. Non, which removes trials with both the Delay and Count ? codes, is the strongest model. The model detecting Count v. Non-count, which removes the Delay codes, is also very strong. Intuitively, clearer labels should result in a stronger model, so this finding is not particularly surprising. The code Delay likely includes students who are actually counting all, counting on, and using derived facts, so including trials labeled as Delay in any model will make it weaker.

With models that keep ambiguous data, the software is better able to distinguish strategies that have strong time indicators, such as count all, which typically takes much longer than other strategies. The computer log is better at detecting counting all than counting on. Since students need to find their answer on the keyboard, the precision of students' response time is not very exact, and responding after counting on may realistically take the same amount of time as a memorized answer or derived fact. Since many of the students in this study are just beginning to

count on, the strategy likely takes more time at first, becoming quicker as students gain practice. Students already have a lot of practice counting all, so their counting time likely does not have as much variation.

Computer Log Features

Major findings. Features that include time indicators are generally the strongest in a given model, especially those models that isolate counting all. For example, the model predicting Count All v. Count On contains only time features and Count All v. Other contains three time features and one software feature. When the ambiguous data are removed, problem and/or software characteristics become more salient. For example, Count v. Non-count, has one time feature and two problem features, and Accurate + Advanced Strategy v. Non has one time feature, two software features, and two problem features.

It is not surprising that time features are the most salient because response time analysis in combination with problem characteristics has traditionally been used in addition strategy research (Siegler, 1987). As part of this study, Siegler analyzed problem characteristics in combination with researcher-coded strategies to predict the oral response time, which was measured using videotape and precise time codes. Using multiple regression, he found the problem characteristic that accounted for the most variance in response time for each strategy. For example, the size of the smaller addend accounted for 86% of the variance in response time for trials in which the student counted on from the larger addend, whereas for counting all, the size of the sum accounted for the most variance (35%). Data mining techniques in the current study help answer the complementary question of what features help predict the strategy, so it is not surprising that time and problem characteristics both contribute to these detection models.

Implications

For researchers. The current study indicates that young children are able to accurately use more advanced strategies after using carefully designed software. In particular the use of certain tools encourages more advanced strategies, both when the tools are present and when their access is delayed as in the timed round, which may indicate that students are internalizing the visual models presented in the tools. Figuring out how to best help students who struggle is an important role for research. Students who score extremely low on a timed number facts assessment seem to benefit the most from extended exposure to the software. However, the number of agents explaining the content did not seem to affect performance.

Students' general inability to accurately describe the strategies they use suggests that researchers should take young children's self-reports of strategy with a grain of salt, always validating those responses with trained researchers. The study also reveals that open-ended questions and one-on-one interviews may be better methods of evaluating students' strategy awareness. Teaching students to be more aware of what strategies are available, when to use particular strategies, and how to use such strategies is a complex endeavor that may require a more intensive intervention or a combination of software and classroom instruction. In the current study, only the high-performers improved in their strategy recognition, indicating that students may need certain prior knowledge to show metastrategic gains from these computer-based lessons and models.

Educational researchers are using data mining to detect many different behaviors, including detecting exploration strategies of undergraduates (Amershi & Conati, 2009) and inquiry strategies of middle-school students (Gobert, Sao Pedro & Baker, 2012) but relatively little of this work focuses on young children. Detecting young children's addition strategies is a

relatively simple proposition because the strategies are clearly defined, follow a well-researched developmental trajectory, and are highly observable. Trained researchers are able to detect strategy with high reliability, for example, Kappa = .91 in the current study and 94% agreement in Siegler (1987).

Compared to the human observers, the current strategy detection models have a lot of room to grow, the best one achieving Kappa = .67 (A' = .896). However, behavior detection models often start out modest and improve over time. An early model detecting gaming the system reported A' = .85 (Baker, 2004), while later versions that incorporated text replays reported A' = .96 (Baker & de Carvalho, 2009).

For designers. Educational software designers can benefit from conducting cognitive research, analyzing particularly important features, such as tools. In the current study, the results showed particular benefits of the count on tool while revealing potential weaknesses in both the doubles and tens tools. This result indicates that designers should carefully consider the use of continuous versus discrete blocks in software tools. The lack of differences in the number of pedagogical agents shows that, for now, designers may decide to teach content using a single or multiple agents. At the same time, other research on agents suggests particular directions for agent design, including the importance of choice (Moreno & Flowerday, 2006), affect (Lester et al, 2007), and demographic characteristics (Baylor & Kim, 2004).

Analyzing the features that surface in the strategy detection models also offers insight into design considerations. Since tools are an important part of the software and are meant to encourage particular strategies, the fact that they do not strongly detect students' strategy indicates that they may need revision. Students' self-report was not included in any model, which

indicates that this feature is not functioning as intended and may need to be redesigned or not included in future iterations of the software.

For teachers. Teachers can learn from this study that software is capable of improving the strategies that young children use. Low-performing students seem to benefit the most in terms of increasing use of advanced strategies, while high-performers improved their metacognitive ability to recognize strategies that another child used. Children may need a certain amount of practice using strategies themselves before they can recognize those same strategies in other children. Teachers should also not necessarily trust students' self-reported strategy, indicating a need to carefully observe students in addition to questioning them about how they solved particular problems.

Detecting strategy also has important practical applications for the classroom. Since students with math learning disabilities tend to use less advanced strategies longer than their classmates (Geary, Hamson, & Hoard, 2000), strategy detection may play a role in screening students to determine if they are at risk of poor math performance. Measures that currently incorporate information about students' strategy use are typically administered one on one, taking away valuable teaching time. Software that includes strategy detection could get at this same information without the hassle of administering time-consuming assessments. At the very least, the software could identify which students really need a more comprehensive assessment.

Strategy detection also provides very practical information to share with teachers. Many assessments give teachers little more than a score and a percentile ranking, leaving teachers to figure out how to intervene. Successful strategy detection could tell those teachers which students are still counting all and would benefit from mini-lessons focused on counting on. Or

the software could identify which students are currently counting on from the first addend and could use a lesson on counting on from the larger addend.

Limitations

One important limitation is that the current study includes a relatively small sample, only 47 students, which limits the power of statistical comparisons. Nearly half of these students (n = 23) would be flagged as at-risk on the basis of their number facts pretest scores, which adds to the possibility that the general results represent regression to the mean. Since the two treatment groups both used the software, differing only in the instructional videos presented, the pre/posttest effects may represent maturation or be due to classroom practice. However, this is not likely the entire story considering that students increased their use of advanced strategies after just one session using the software.

Because the Addition Strategy Awareness measure was developed concurrent to the intervention, three of the four subtests were only delivered at posttest, limiting the ability to show growth. The instructional videos teaching about the four different strategies were also quite short, each less than a minute. Compared to the use of the software, which lasted 10 - 15 minutes, the instruction may have been too short and too infrequent. A follow up study might investigate the effects of more instructional videos.

The forward selection process used to distill the detecting models also has an important limitation. Using too many features in a model creates a greater possibility of fitting to the noise in a data set, which is why forward selection can be useful. However, using Kappa and AUC values to decide what features to keep in a model has the potential to create another type of overfit, selecting variables that are particularly sensitive to this particular sample, which may also limit future generalizability of the models with new data sets.

Since the strategy detection models that remove ambiguous data are more powerful, special attention should be paid to evaluate whether trials with ambiguous strategy codes differ in substantive ways from those with explicit strategies. The Internal v. Visible contrast did not produce a very strong model, which indicates that students' use of an internal strategy does not strongly affect their actions in the software.

Future Directions

There are many areas of future research given the results of the current study. With the current MathemAntics activity, future studies could evaluate tool revisions, directly testing the hypothesis that continuous boxes lead to more advanced strategies than discrete boxes. Working with both younger and older children could reveal different patterns of strategy emergence. For example, older children who have already mastered counting on may be better primed to learn about the derived facts strategy. Older children may also be better at reporting the strategy they used to the computer.

To better investigate the effects of multiple agents on addition strategy awareness, researchers could test similar instructional videos presented separately from the rest of the activity with both first graders as well as older children. Such a study might indicate that the concentration of instruction was not sufficient in the current study. Future research with the multiple agents should also include measures of character like-ability to ensure that inadvertent character design decisions were not influencing children's ability to learn from particular characters.

Looking at future directions in data mining with the existing data set, the strategy detection models may benefit from more feature engineering, creating new features calculated from existing features or combinations of features. It may be that a multiplicative interaction of

two features may be more powerful than each feature individually. For example, multiplying trial speed by historical speed would further distinguish those who are both fast on this trial and fast historically and those who are slow on both, while keeping those who are slow on only one or the other as relatively equivalent. It may also be that the standard deviation of a particular feature may be more predictive than the mean.

Also with the existing data, it would be interesting to create Bayesian Knowledge Tracing (BKT) models to predict the probability that a particular student has learned a strategy at a particular trial. For this type of analysis it would be important to use a Strategy v. Other contrast, which includes all ambiguous data. Using Count All v. Other in combination with Accuracy, a BKT model could predict the likelihood that a student has learned to use a non-counting-all strategy correctly at a given trial.

Collecting future data could indicate whether improving the tools makes the tools more likely to add to a given strategy detection model. Adding a more comprehensive, standardized screening measure to the pre and posttest could also directly test the hypothesis that playing with such software could identify students who are at risk of future math difficulty. The research question might ask: For how long do students need to play with the software to achieve a reasonable reliability with the standardized screening measure scores?

Because researchers worked one on one with students as they played with the software, the current study was limited in the number of students and amount of time working with each student. Using a protocol adapted from Ocumpaugh, Baker and Rodrigo (2012), researchers could instead observe multiple students as they use the software. The researcher would observe students for a period of 10 - 20 seconds, coding observed behaviors on an Android device that is time-synced with the computer logging system. This protocol allows a single researcher to

observe many more students, conduct the research within the classroom (as opposed to pulling students out of the class), and have less influence over student behavior, mimicking a more typical classroom implementation. Conducting research in this way would allow us to better answer both specific behavior detection questions as well as more general cognitive psychology investigations.

References

Aleven, V., & Koedinger, K. R. (2002). An effective metacognitive strategy: learning by doing and explaining with a computer-based Cognitive Tutor. *Cognitive Science*, *26*(2), 147–179.

Amershi, S., & Conati, C. (2009). Combining unsupervised and supervised classification to build user models for exploratory learning environments. *Journal of Educational Data Mining*, *1*(1), 1–54.

Baker, R. S. J. D., Corbett, A. T., Roll, I., & Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. In *User Modeling and User-Adapted Interaction* (pp. 1–36).

Baker, R. S. J. D., Kalka, J., Aleven, V., Rossi, L., Gowda, S., Wagner, A., … Ocumpaugh, J. (2012). Towards sensor-free affect detection in cognitive tutor algebra. In *Educational Data Mining*.

Baker, R. S. J. D., & Yacef, K. (2009). The state of educational data mining in 2009: A review and future visions. *Journal of Educational Data Mining*.

Baroody, A. J. (2003). The development of expertise and flexibility: The integration of conceptual an procedural knowledge. In A. J. Baroody & A. Dowker (Eds.), *The Development of Arithmetic Concepts and Skills: Constructing Adaptive Expertise* (pp. 1–33). Mahwah, NJ: Lawrence Erlbaum Associates.

Baylor, A. L. (2002). Expanding preservice teachers' metacognitive awareness of instructional planning through pedagogical agents. *Educational Technology Research and Development*, *50*(2), 5–22.

Baylor, A. L., & Chang, S. (2002). Pedagogical agents as scaffolds: The role of feedback timing, number of agents, and adaptive feedback. In *International Conference of the Learning Sciences, Seattle, WA* (pp. 3–4).

Baylor, A. L., & Ebbers, S. (2003). The pedagogical agent split-persona effect: When two agents are better than one. In *ED-MEDIA* (pp. 1–5). Honululu, Hawaii.

Baylor, A. L., & Kim, Y. (2004). Pedagogical agent design: The impact of agent realism, gender, ethnicity, and instructional role. *Intelligent Tutoring Systems*, (1997).

Baylor, A. L., & Kim, Y. (2005). Simulating instructional roles through pedagogical agents. *International Journal of Artificial Intelligence in Education*, *15*(2), 95–115.

Baylor, A. L., & Ryu, J. (2003). The effects of image and animation in enhancing pedagogical agent persona. *Journal of Educational Computing Research*, *28*(4), 373–394.

Beck, J. E., Jia, P., Sison, J., & Mostow, J. (2003). *Predicting student help-request behavior in an intelligent tutor for reading*. *User Modeling 2003* (pp. 303–312). Springer Berlin Heidelberg.

Beck, J. & Mostow, J. (2008). How who should practice: Using learning decomposition to evaluate the efficacy of different types of practice for different types of students. In *Proceedings of the 9th International Conference on Intelligent Tutoring Systems* (pp. 353–362).

Bottino, R. M., & Chiappini, G. (2002). Advanced technology and learning environments: Their relationships within the arithmetic problem-solving domain. In *Handbook of International Research in Mathematics Education* (pp. 757–785).

Carpenter, T. P., & Moser, J. M. (1984). The acquisition of addition and subtraction concepts in grades one through three. *Journal for Research in Mathematics Education*, *15*(3), 179–202.

Carver, S. M., & Klahr, D. (1986). Assessing children's LOGO debugging skills with a formal model. *Journal of educational computing research*, *2*(4), 487-525.

Cetintas, S., Si, L., Xin, Y., Hord, C., & Zhang, D. (2009). Learning to identify students' off-task behavior in intelligent tutoring systems, 2–4.

Clements, D. H., & Battista, M. T. (2000). Designing effective software. In Kelly R. A. & A. E. Lesh (Eds.), *Handbook of Research Design in Mathematics and Science Education* (pp. 761–776). Mahway, NJ: Lawrence Erlbaum Associates.

Cowan, R. (2003). Does it all add up? Changes in children's knowledge of addition combinations, strategies, and principles. In A. J. Baroody & A. Dowker (Eds.), *The Development of Arithmetic Concepts and Skills: Constructing Adaptive Experience* (Vol. 1890, pp. 35–74). Mahwah, NJ: Lawrence Erlbaum Associates.

Cross, C. T., & Woods, T. A. (2009). The teaching-learning paths for number, relations, and operations. In C. T. Cross, T. A. Woods, & H. Schweingruber (Eds.), *Mathematics Learning in Early Childhood: Paths Toward Excellence and Equity*. Washington DC: Committee on Early Childhood Mathematics; National Research Council and National Academy of Sciences.

Csíkszentmihályi, M. (1991). *Flow: the psychology of optimal experience*. New York: HarperCollins.

D'Mello, S., & Graesser, A. (2009). Automatic detection of learner's affect from gross body language. *Applied Artificial Intelligence*, *23*(2), 123–150.

Edwards, L. (1998). Embodying mathematics and science: Microworlds as representations. *The Journal of Mathematical Behavior*, *17*(1), 53–78.

Efron, B., & Gong, G. (1983). A leisurely look at the bootstrap , the jackknife, and cross-validation. *The American Statistician*, *37*(1), 36–48.

Flavell, J. H. (1979). Metacognition and cognitive monitoring: A new area of cognitive-developmental inquiry. *American Psychologist*, *34*(10), 906–911.

Fuson, K. C., Kalchman, M., & Bransford, J. D. (2005). Mathematical understanding: An introduction. In J. Bransford & M. Donovan (Eds.), *How Students Learn: History, Mathematics, and Science in the Classroom* (pp. 217–256). Washington, DC: National Academy Press.

Geary, D. C., Hamson, C. O., & Hoard, M. K. (2000). Numerical and arithmetical cognition: a longitudinal study of process and concept deficits in children with learning disability. *Journal of experimental child psychology*, *77*(3), 236–63.

Geary, D. C., Hoard, M. K., Byrd-Craven, J., & DeSoto, M. C. (2004). Strategy choices in simple and complex addition: Contributions of working memory and counting knowledge for children with mathematical disability. *Journal of experimental child psychology*, *88*(2), 121–51.

Gee, J. P. (2005). Learning by design: Good video games as learning machines. *E-Learning and Digital Media*, *2*(1), 5–16.

Ginsburg, H. P. (1989). *Children's arithmetic* (2nd ed.). Austin, TX: Pro-Ed.

Ginsburg, H. P., Carpenter, K. K., & Labrecque, R. (2011). Introduction to MathemAntics: Software for children from age 3 to grade 3. In *International Society for Design and Development Conference*. Boston, MA.

Ginsburg, H. P., Chiong, C., Lee, Y., & Pappas, S. (2010). The clinical interview should supplement or perhaps replace CBM progress monitoring: The case of early mathematics.

Ginsburg, H. P., & Pappas, S. (2004). SES, ethnic, and gender differences in young children's informal addition and subtraction: A clinical interview investigation. *Journal of Applied Developmental Psychology*, *25*, 171–192.

Gobert, J., Pedro, M. S., & Baker, R. S. J. D. (2012). Leveraging educational data mining for real time performance assessment of scientific inquiry skills within microworlds. *Journal of Educational Data Mining*, *4*(1).

Gray, E. M. (1991). An analysis of diverging approaches to simple arithmetic: Preference and its consequences. *Educational Studies in Mathematics*, *22*(6), 551–574.

Gray, E. M., & Tall, D. (1994). Duality, ambiguity, and flexibility: A "proceptual" view of simple arithmetic. *Journal for Research in Mathematics Education*, *25*(2), 116–140.

Gulz, A. (2004). Benefits of virtual characters in computer based learning environments: Claims and evidence. *International Journal of Artificial Intelligence in Education*, *14*(3, 4), 313–334.

Handler Miller, C. (2008). *Digital Storytelling:* (2nd ed., p. 496). Burlington, MA: Focal Press.

Hanley, J. a, & McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, *143*(1), 29–36.

Hoyles, C., & Noss, R. (2009). The technological mediation of mathematics and its learning. *Human Development*, *52*, 129–147.

Jeong, H., & Biswas, G. (2008). Mining student behavior models in learning-by-teaching environments. *International Conference on Educational Data Mining*, 127–136.

Jordan, N. C., Hanich, L. B., & Kaplan, D. (2003). Arithmetic fact mastery in young children: A longitudinal investigation. *Journal of Experimental Child Psychology*, *85*(2), 103–119.

Jordan, N. C., Kaplan, D., Nabors Oláh, L., & Locuniak, M. N. (2006). Number sense growth in kindergarten: a longitudinal investigation of children at risk for mathematics difficulties. *Child development*, *77*(1), 153–75.

Jordan, N. C., & Levine, S. C. (2009). Socioeconomic variation, number competence, and mathematics learning difficulties in young children. *Developmental disabilities research reviews*, *15*(1), 60–8.

Kiili, K. (2005). Digital game-based learning: Towards an experiential gaming model. *The Internet and Higher Education*, *8*(1), 13–24.

Kuhn, D. (1995). Microgenetic Study of Change: What Has It Told Us? *Psychological Science*, *6*(3), 133–139.

Kuhn, D. (2000). Metacognitive Development. *Current Directions in Psychological Science*, *9*(5), 178–181.

Lee, Y.-S., Pappas, S., Chiong, C., & Ginsburg, H. P. (2010). *mCLASS®: MATH - Technical manual*. Brooklyn, NY: Wireless Generation.

Lester, J. C., Converse, S. A., Kahler, S. E., Barlow, S. T., Stone, B. A., & Bhogal, R. S. (1997). The persona effect: affective impact of animated pedagogical agents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 359–366). ACM.

Malone, T. W. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science*, *5*(4), 333–369.

Malone, T. W., & Lepper, M. R. (1987). Making learning fun: A taxonomy of intrinsic motivations for learning. In R. E. Snow & M. J. Farr (Eds.), *Aptitude, Learning, and*

Instruction: Vol. 3. Conative and Affective Process Analysis (pp. 223–253). Hillsdale, NJ: Lawrence Erlbaum Associates.

Mavrikis, M. (2008). Data-driven modeling of students' interactions in an ILE. In *The 1st International Conference on Educational Data Mining* (pp. 87–96). Montreal, Canada.

Moreno, R., & Flowerday, T. (2006). Students' choice of animated pedagogical agents in science learning: A test of the similarity-attraction hypothesis on gender and ethnicity. *Contemporary Educational Psychology*, *31*(2), 186–207.

Moreno, R., Mayer, R. E., Spires, H. A., & Lester, J. C. (2001). The case for social agency in computer-based teaching: Do students learn more deeply when they interact with animated pedagogical agents? *Cognition and Instruction*, *19*(2), 177–213.

Papert, S. (1993). Microworlds : Incubators for Knowledge (pp. 120–134).

Pearson/Scott Foresman, TERC (Firm), & Pearson Education, Inc. (2008). *Investigations in number, data, and space*. Pearson Scott Foresman.

Putnam, R. T., De Bettencourt, L. U., & Leinhardt, G. (1990). Understanding of derived-fact strategies in addition and subtraction. *Cognition and instruction*, *7*(3), 245–285.

Quintana, C., Shin, N., Norris, C., & Soloway, E. (2006). Learner-centered design: Reflections on the past and directions for the future. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 119–134). Cambridge: Cambridge University Press.

Randolph, J. (2005). Multi-rater, free-marginal variation of the Fleiss Kappa: An alternative to Fleiss' fixed-marginal multi-rater kappa. In *Joensuu University Learning and Instruction Symposium*.

Reeves, B., & Nass, C. (1999). *The media equation: How people treat computers, television, and new media like real people and places*. *International Journal of Instructional Media*. Cambridge: Cambridge University Press.

Rittle-Johnson, B., Siegler, R. S., & Alibali, M. W. (2001). Developing conceptual understanding and procedural skill in mathematics: An iterative process. *Journal of educational psychology*, *93*(2), 346-362.

Romero, C., & Ventura, S. (2007). Educational data mining: A survey from 1995 to 2005. *Expert Systems with Applications*, *33*(1), 135–146.

Sao Pedro, M. A., d Baker, R. S., & Gobert, J. D. (2012). Improving construct validity yields better models of systematic inquiry, even with less information. In *User Modeling, Adaptation, and Personalization* (pp. 249-260). Springer Berlin Heidelberg.

Sarama, J., & Clements, D. H. (2002). Building Blocks for young children's mathematical development. *Journal of Educational Computing Research*, *27*(1), 93–110.

Schell, J. (2010). Transmedia worlds. In *Dust or Magic: Children's New Media Institute*. Lambertsville, NJ.

Secada, W., Fuson, K. C., & Hall, J. W. (1983). The transition from counting-all to counting-on in addition. *Journal for Research in Mathematics Education*, *14*(1), 47–57.

Shrager, J., & Siegler, R. S. (1998). SCADS: A model of children's strategy choices and strategy discoveries. *Psychological Science*, *9*(5), 405.

Shute, V. (2011). Stealth assessment in computer-based games to support learning. In *Computer Games and Instruction* (pp. 503–524). Charlotte, NC: Information Age Publishing.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example From children's addition. *Journal of Experimental Psychology*, *116*(3).

Siegler, R. S. (1988). Individual differences in strategy choices: good students, not-so-good students, and perfectionists. *Child development*, *59*(4), 833–51.

Siegler, R. S. (2007). Cognitive variability. *Developmental science*, *10*(1), 104–9.

Siegler, R. S., & Crowley, K. (1994). Constraints on learning in nonprivileged domains. *Cognitive Psychology*.

University of Chicago School Mathematics Project. (2007). *Everyday Mathematics*. Chicago: Wright Group/McGraw-Hill.

Vygotsky, L. S. (1978). Interaction between learning and development. In *Mind and Society*. Cambridge, MA: Harvard University Press.

White, B. Y., Shimoda, T. A., & Fredericksen, J. R. (2000). Facilitating students' inquiry learning and metacognitive development through modifiable software advisers. In S. Lajoie (Ed.), *Computers as Cognitive Tools Vol 2: No More Walls* (pp. 97–132). Mahwah, NJ: Lawrence Earlbaum.